

Computing Strong and Weak Permissions in Defeasible Logic

Guido GOVERNATORI^a, Francesco OLIVIERI^{a,b,c}
Antonino ROTOLO^d and Simone SCANNAPIECO^{a,b,c}

^a*NICTA, Queensland Research Laboratory, Australia*

^b*Department of Computer Science, University of Verona, Italy*

^c*Institute for Integrated and Intelligent Systems, Griffith University, Australia*

^d*CIRSFID, University of Bologna, Italy*

Abstract In this paper we propose an extension of Defeasible Logic to represent and compute three concepts of defeasible permission. In particular, we discuss different types of explicit permissive norms that work as exceptions to opposite obligations. Moreover, we show how strong permissions can be represented both with, and without introducing a new consequence relation for inferring conclusions from explicit permissive norms. Finally, we illustrate how a preference operator applicable to contrary-to-duty obligations can be combined with a new operator representing ordered sequences of strong permissions which derogate from prohibitions. The logical system is studied from a computational standpoint and is shown to have linear computational complexity.

1. Introduction

The concept of permission plays an important role in many normative domains in that it may be crucial in characterising notions such as those of authorisation and derogation [11,30,33]. For example, sometimes it may happen that we mistakenly drive to a building site, or a road-work restricted area, with signs out saying “No admittance. Authorised personnel only”. Or consider when we subscribe to an on-line sale agreement accepting to enter our personal data on the condition that this information is only used for shipping, and other necessary purposes to communicate with us or deliver the products to us. In both cases, a permission (to enter a restricted area or to use our personal data) is stated as an exception to a general prohibition.

Despite this fact, the concept of permission is still elusive in this field of literature and has not been extensively investigated in deontic logic as the notion of obligation. For a long time, deontic logicians mostly viewed permission as the dual of obligation: $Pa \equiv \neg O\neg a$. This view is unsatisfactory, as it hardly allows us to grasp the meaning of examples like the ones previously mentioned. For this, and other reasons, the attempt to reduce permissions to duals of obligations has been criticised (see [2,1]).

One important distinction that has traditionally contributed to a richer account of this concept is the one between *weak* (or *negative*) and *strong* (or *positive*) permission [35]. The former corresponds to saying that some a is permitted if $\neg a$ is not provable

as mandatory. In other words, something is allowed by a code iff(only when) it is not prohibited by that code. At least when dealing with unconditional obligations, the notion of weak permission is trivially equivalent to the dual of obligation [25].

The latter concept of strong permission is more complicated, as it amounts to saying that some a is permitted by a code iff such a code explicitly states that a is permitted. It follows that a strong permission is not derived from the absence a prohibition, but is explicitly formulated in a permissive norm. The complexities of this concept depend on the fact that, besides “the items that a code explicitly pronounces to be permitted, there are others that in some sense follow from the explicit ones”. The problem is hence “to clarify the inference from one to the other” [25, p. 391–2]. For example, if some b logically follows from a , which is strongly permitted, can we say that b is also strongly permitted?

Features such as the distinction between strong and weak permission show the multifaceted nature of permission and permissive norms, which has been overlooked by most logicians for a long time. Nevertheless, some exceptions have recently offered significant contributions to the logical understanding of permission [25,9,10,12,33,32]. These contributions can be roughly summarised into the following points:

- despite some scepticism [28,27] and critical remarks [3,2] (a discussion of this related work can be found in Section 7), the distinction between weak and strong permission seems to be needed, otherwise it is rather hard to account for the fact that certain permissions make sense because they explicitly derogate to *existing* prohibitions while other permissions are not explicit and occur *precisely* because opposite prohibitions *do not exist*;
- we may have different types of strong permissions (specifically permissions that logically follow from explicit permissive norms), according to whether
 - * we statically determine what is actually permitted given what is obligatory and what is explicitly permitted;
 - * we dynamically determine “the limits on what may be prohibited without violating static permissions” [9];
- especially in the *law*, strong permissions state exceptions to obligations [8]: indeed, derogating with a permission, for example, to a general prohibition to use private protected data provides an exception to such a prohibition;
- strong permissions make sense even when any incompatible prohibitions are not in the legal system; permissions have a dynamic behaviour and prevent future prohibitions from holding in general, or applying to specific contexts [13].

This paper moves from the above points with the specific purpose of studying the different conceptual and computational aspects of weak and strong permissions. More precisely, the current contribution works in the following directions:

Permissions and defeasibility. The concept of permission exhibits strong connections with the idea of defeasibility. Indeed, an example of a natural way to capture strong permissions acting as exceptions to obligations is the one where permissions rebut the conclusions of incompatible prescriptive norms [25,9,17,33] or undercut them (i.e., challenge an inference rule of an argument supporting an opposite obligation) [10]).

Permissions and preferences. Sometimes explicit derogations of (existing or possible) prohibitions can be ranked according to some preference orderings. In other words, given any prescriptive norm prohibiting a , more derogations to this norm can be stated and ranked in a certain preference sequence. This situation may occur in domains such as the law, where for instance the lawmaker, when imposing duties for citizens, establishes conditions to lessen the effect of violating such duties to different degrees, or exempt people to comply with the duties. We will study these mechanisms and see that ordered sequences of strong permissions, derogating or making exceptions to prohibitions, have interesting similarities with ordered sequences of contrary-to-duty obligations [14,15]. This is a specific novelty of our contribution, as such sequences regard permissions (i.e., exceptions) which are not necessarily incompatible with each other.

Permissions and computation. If, as we have mentioned, the concept of permission is mostly overlooked in literature, being that its computational treatment is basically neglected. To the best of our knowledge, no work in deontic logic has extensively explored the computational complexity of reasoning about different types of permission. Here, we will attempt a first analysis of the problem in the context of a modal extension of Defeasible Logic [5]. Modal Defeasible Logic is a computationally efficient logical framework able to capture various aspects of non-monotonic and modal reasoning, as well as the defeasible character of permissive norms, and recently a possible-world semantics for it has been proposed [18]. We will study how to compute weak and strong permissions with and without introducing a new non-monotonic consequence relation for permission. The choice of Defeasible Logic is motivated by the fact that it is very efficient. Also, the formal language of its (non-modal) modulo is simple, thus allowing us to isolate the deontic aspects of permissions and investigate their specific computational characteristics.

The layout of the paper is as follows. Section 2 introduces and informally discusses three types of defeasible permission in Defeasible Logic. Section 3 presents the technical machinery and states coherency and consistency results of the proposed extension. In Section 4 we develop the algorithmic means to state what is mandatory and what is permitted in a given theory, along with the corresponding computational results in Section 5. Section 6 discusses the system and illustrates how the logical framework presented in Section 3 is able to capture the three types of permission. Section 7 discusses some related work and provides a summary of the paper.

2. Three concepts of permission

This section is meant to offer a brief and gentle introduction to our formal language and logic, and to discuss three different types of permission and their relation with the concept of normative defeasibility. Moreover, we illustrate the idea of preference over permissions that explicitly derogate to prohibitions.

These aspects will be formally handled in Section 3. The whole discussion of the computational aspects of permission in Defeasible Logic is postponed to Sections 4 and 5.

2.1. Informal presentation of the logic

Let us summarise the basic logical intuitions behind our framework.

1. Permissive and prescriptive norms are represented by means of defeasible rules, whose conclusions normally follow unless they are defeated by contrary evidence. For example, the rule

$$Order \Rightarrow_O Pay$$

says that, if we send a purchase order, then we will be defeasibly obliged to pay; the rule

$$Order, Creditor \Rightarrow_P \neg Pay$$

states that if we send an order, in general we are not obliged to pay if we are creditors towards the vendor for the same amount.

2. Rules introduce modalities: if we have the rule $a \Rightarrow_O b$ and a holds, then we obtain Ob . That is to say, in the scenario where conditions described by a hold, the obligation of doing b is active as well. The advantage is that explicitly deriving modal literals such as Ob adds expressive power to the language, since Ob may appear in the antecedent of other rules, which can then be triggered.
3. For the sake of simplicity, modal literals can only occur in the antecedent of rules. In other words, we do not admit nested modalities, i.e., rules such as $a \Rightarrow_O Pb$. This is in line with our idea that the applicability of rules labeled with modality \Box (where \Box can be O for obligation or P for permission) is the condition for deriving literals modalised with \Box .
4. The symbols O and P are not simple labels: they are modalities. O is non-reflexive¹: consequently, we do not have a conflict within the theory when $\neg a$ is the case and we derive that a is mandatory (Oa); this amounts to having a violation. The modality P works in such a way that two rules for P supporting a and $\neg a$ do not clash, but a rule like $\Rightarrow_P b$ attacks a rule such as $\Rightarrow_O \neg b$ (and vice versa).
5. Like standard Defeasible Logic, our extension is able to establish the relative strength of any rule (thus to solve rule conflicts) and has two types of attackable rules: defeasible rules and defeaters. Defeaters in Defeasible Logic are a special kind of rules: they are used to prevent conclusions but not to support them. For example, the defeater

$$SpecialOrder, PremiumCustomer \rightsquigarrow_O \neg PayBy7Days$$

can prevent the derivation of the obligation for premium customers placing special orders to pay within the deadline of 7 days, but cannot be used to directly derive any conclusion.

2.2. Permissions and defeasibility

The above framework, though simple, allows us to express three basic types of permissions as well as illustrate interesting connections with the idea of defeasibility.

¹As it is well-known, in a non-reflexive modal logic $\Box a$ does not imply a , where \Box is a modal operator.

Weak permission. A first way to define permissions in Defeasible Logic is by simply considering weak permissions and stating that the opposite of what is permitted is not provable as obligatory. Let us consider a normative system consisting of the following two rules:

$$\begin{aligned} r_1 &: \text{Park}, \text{Vehicle} \Rightarrow_O \neg \text{Enter} \\ r_2 &: \text{Park}, \text{Emergency} \Rightarrow_O \text{Enter}. \end{aligned}$$

Here the normative system does not contain any permissive norm. However, since Defeasible Logic is a sceptical non-monotonic logic, in case both r_1 and r_2 fire we neither conclude that it is prohibited nor that it is obligatory to enter, because we do not know which rule is stronger. Hence, in this context, both $\neg \text{Enter}$ and Enter are weakly permitted.

As already argued, this is the most direct way to define the idea of weak permission: some q is permitted by a code iff q is not prohibited by that code. Accordingly, saying that any literal q is weakly permitted corresponds to the failure of deriving $\neg q$ using rules for O. Notice that, in Defeasible Logic, this does not amount to obtain $\neg O \neg q$.

Explicit permissions are defeaters. In Defeasible Logic any rule can be used to prevent the derivation of a conclusion. For instance, suppose there exists a norm that prohibits to U-turn at traffic lights unless there is a “U-turn permitted” sign:

$$\begin{aligned} r_1 &: \text{AtTrafficLight} \Rightarrow_O \neg \text{Uturn} \\ r_2 &: \text{AtTrafficLight}, \text{UturnSign} \Rightarrow_O \text{Uturn}. \end{aligned}$$

In this example we use a defeasible rule for obligation to block the prohibition to U-turn. However, this is not satisfactory: if we do not know whether r_2 is stronger than r_1 , then the best we can say is that U-turn is weakly permitted. Furthermore, if r_2 prevails over r_1 , we derive that U-turn is obligatory.

Thus, there are good reasons to argue that defeaters for O are suitable to express an idea of strong permission². Explicit rules such as $r : a \rightsquigarrow_O q$ state that a is a specific reason for blocking the derivation of $O \neg q$ (but not for proving Oq). In other words, this rule does not support any conclusion, but states that $\neg q$ is deontically undesirable. Consider this example:

$$\begin{aligned} r_1 &: \text{Weekend}, \text{AirPollution} \Rightarrow_O \neg \text{UseCar} \\ r_2 &: \text{Weekend}, \text{Emergency} \rightsquigarrow_O \text{UseCar}. \end{aligned}$$

Rule r_1 states that on weekends it is forbidden to use private cars if a certain air pollution level is exceeded. Defeater r_2 is in fact an exception to r_1 , and so it seems to capture the above idea that explicit permissive norms (especially in the law) provide exceptions to obligations.

Explicit permissions using permissive rules. Another approach is based on introducing specific rules for deriving permissions [25,9]. Let us consider the following situation:

$$\begin{aligned} r_1 &: \text{Weekend}, \text{AirPollution} \Rightarrow_O \neg \text{UseCar} \\ r'_2 &: \text{Emergency} \Rightarrow_P \text{UseCar}. \end{aligned}$$

As r_2 in the previous scenario, r'_2 looks like an exception to r_1 . The apparent difference between r_2 and r'_2 is that the latter is directly used to prove that the use of the car is permitted ($P\text{UseCar}$) in case of emergencies. The question is: does it amount to a real difference?

²The idea of using defeaters to introduce permissions was introduced in [19].

Although r_2 is a defeater, it is specifically used to derive the strong permission to use the car, like r'_2 . In addition, rules such as r'_2 do not attack other permissive rules, but are in conflict only with rules for obligation intended to prove the opposite conclusion. This precisely holds for defeaters.

Moreover, let us suppose to have the defeater $s : a \rightsquigarrow_P b$. Does s attack a rule like $\Rightarrow_P \neg b$?

If this is the case, s would be close to an obligation. The fact that Pb does not attack $P\neg b$ makes it pointless for s to introduce defeaters for P . But, if this is not the case, s could only attack $\Rightarrow_O \neg b$, thus being equivalent to $s' : a \rightsquigarrow_O b$.

Therefore, although it is admissible to have defeaters, we do not need to distinguish defeaters for O from those for P . One way to mark the difference between \rightsquigarrow and \Rightarrow_P is by stating that only the latter rule type admits ordered sequences of strong permissions in the head of a rule, which are supposed to derogate or make exceptions to prohibitions. This matter will be discussed in the next subsection.

2.3. Permissions, obligations, and preferences

The introduction of ordered sequences of strong permissions in the head of a rule, which derogate or make exceptions to prohibitions, can be logically modelled by enriching the formal language and following these guidelines:

1. In many domains, such as the law, norms often specify mandatory actions to be taken in case of their violation. In general, obligations in force after the violation of some other obligations correspond to contrary-to-duty (CTD) obligations. These constructions affect the formal characterisation of compliance since they identify situations that are not ideal, but still acceptable. A compact representation of CTDs may resort to the non-boolean connective \otimes [14]: a formula like $x \Rightarrow_O a \otimes b$ means that, if x is the case, then a is obligatory, but if the obligation a is not fulfilled, then the obligation b is activated and becomes in force until it is satisfied, or violated.
2. Concepts introduced at point 1 can be extended to permissive rules with the subscripted arrow \Rightarrow_P by introducing the non-boolean connective \odot for sequences of permissions. As in the case of \otimes , given a rule like $\Rightarrow_P a \odot b$, we can proceed through the \odot -chain to obtain the derivation of Pb . However, permissions cannot be violated, and consequently it does not make sense to obtain Pb from $\Rightarrow_P a \odot b$ and $\neg a$. In this case, the reason to proceed in the chain is rather that the normative system allows us to prove $O\neg a$. Hence, \odot still establishes a preference order among strong permissions and, in case the opposite obligation is in force, another permission holds. This is significant especially when strong permissions are exceptions to obligations.

In this paper we take a neutral approach as to whether ordered sequences of obligations or permissions are either given explicitly, or inferred from other rules. However, we point out that normative documents often explicitly contains provision with such structures. A clear example of this is provided by the Australian “National Consumer Credit Protection Act 2009” (Act No. 134 of 2009) which is structured in such a way that for every section establishing an obligation or a prohibition, the penalties for violating the provision are given in the section itself.

Example 1 (National Consumer Credit Protection Act 2009). *Section 29 (Prohibition on engaging in credit activities without a licence) of the act recites:*

(1) A person must not engage in a credit activity if the person does not hold a licence authorising the person to engage in the credit activity.

Civil penalty: 2,000 penalty units.

[...]

Criminal penalty: 200 penalty units, or 2 years imprisonment, or both.

This norm can be represented as

$$r_1 : \Rightarrow_O \neg \text{CreditActivity} \otimes 2000 \text{CivilPenaltyUnits}$$

$$r_2 : \text{CreditLicence} \Rightarrow_P \text{CreditActivity}$$

where $r_2 > r_1$. The first rules state that in absence of other information a person is forbidden to engage in credit activities ($O \neg \text{CreditActivity}$), and then the second rule establish an exception to the prohibition, or in other terms it recites a condition under which such activities are permitted. The section then continues by giving explicit exceptions (permissions) to the prohibition to engage in credit activity, even without a valid licence.

Sequences of permissions are a natural fit for expressions like “the subject is authorised, in order of preference, to do the following: (list)” or “the subject is entitled, in order of preference, to one of the following: (list)”. This is illustrated in the next example.

Example 2 (U.S. Copyright Act). A concrete instance of sequences of permissions is given by Section 504(c)(1) (Remedies for infringement: Damages and profits) of the U.S. Copyright Act (17 USC §504).

Except as provided by clause (2) of this subsection, the copyright owner may elect, at any time before final judgment is rendered, to recover, instead of actual damages and profits, an award of statutory damages for all infringements involved in the action, with respect to any one work, for which any one infringer is liable individually, or for which any two or more infringers are liable jointly and severally, in a sum of not less than \$750 or more than \$30,000 as the court considers just. [...]

The above provision can be modelled as

$$\text{infringement, beforeJudgment} \Rightarrow_P \text{ActualDamages} \odot \text{StatutoryDamages}$$

The above rendering of the textual provision is based on the interpretation of the term ‘instead’, which suggests that the copyright owners are entitled by default the award of the actual damages and profits, but they may elect to recover statutory damages, which is then the second option if exercised by the relevant party.³

As we have just seen, chains of obligations are appropriate to capture the obligations and the penalties related to them. Furthermore, this kind of structure has been successfully used for applications in the area of business process compliance [21]. In a situation governed by the rule $\Rightarrow_O a \otimes b$ and where $\neg a$ and b hold, the norm has been complied with (even if to a lower degree than if we had a). On the contrary, if we had two rules $\Rightarrow_O a$ and $\neg a \Rightarrow_O b$, then the first norm would have been violated, while the second would have been complied with. But in overall, the whole case would be not compliant [20].

³Here we speak of *entitlements* or *rights*. A *right* is a permission on one party (in this case the copyright owner) generating an obligation on another party (in this case the infringer). For a more detailed discussion on the concept of right see [30].

Consider the following example:

$$\begin{aligned} r_1 &: \text{Invoice} \Rightarrow_O \text{PayWithin7days} \\ r_2 &: \text{OPayWithin7days}, \neg \text{PayWithin7days} \Rightarrow_O \text{Pay5\%Interest} \\ r_3 &: \text{OPay5\%Interest}, \neg \text{Pay5\%Interest} \Rightarrow_O \text{Pay10\%Interest}. \end{aligned}$$

What happens if a customer violates both the obligation to pay within 7 days after the invoice and the obligation to pay the 5% of interest, but she pays the total amount plus the 10% of interest? In the legal perspective the customer should be still compliant, but in this representation contract clauses r_1 and r_2 have been violated. However, if we represent the whole scenario with the single rule

$$\text{Invoice} \Rightarrow_O \text{PayBy7days} \otimes \text{Pay5\%Interest} \otimes \text{Pay10\%Interest},$$

then the rule is not violated, and the customer is compliant with the contract.

Even when the text of legal provisions does not explicitly have this form, there are cases where the joint interpretation of several legal provisions still leads to formulate applicable norms with orders among derogations.

Example 3 (Formal equality and affirmative action). *Art. 3, 1st paragraph, of the Italian constitution ensures formal equality of citizens (in fact, all individuals) before the law, namely, an equal legal treatment for everybody:*

All citizens have equal social dignity and are equal before the law, without distinction of sex, race, language, religion, political opinion, personal and social conditions. [...]

This general principle can be sometimes derogated, for example, when derogations are meant “to remove those obstacles of an economic or social nature which constrain the freedom and equality of citizens” (art. 3, 2nd par.). In fact, one may argue that permitting (which is different from imposing as mandatory) the adoption of affirmative action policies in favour of women (e.g., introducing quotas for women in politics and the job market) is a flexible legal measure to remove some of those obstacles. Now, suppose a quota for women is guaranteed in public institutions in hiring and promoting employees, but another similar derogation can be applied to disabled people. Imagine that, in a specific case, it is not possible to apply both derogations (for example, this would lead to exceeding the number of jobs available) and so we have to choose to hire a woman or a disabled man. In absence of any further legal provision, one possible solution is to balance both options with respect to the specific facts X of the case, thus ranking, in a rule r , these options in order of preference, given the facts X (on balancing, see [4,31]). For instance, if disabled men should be favoured over non-disabled women (because disability in this case reinforces a more serious discrimination or disadvantage) then r is the following:

$$r : X \Rightarrow_P \text{Hire_Disabled_Men} \odot \text{Hire_NonDisabled_Women}$$

The reason why we should still keep as a second option Hire_NonDisabled_Women depends on the fact that we can draw only defeasibly the permission of Hire_Disabled_Men. Indeed, we have only considered art. 3 of the Italian constitution but other legal provisions or factual reasons could block this conclusion. For example, suppose that the disabled person applying for the job was some years earlier convicted of the crime of belonging to a mafia organisation, while the law prohibits in general and without exceptions for public institutions to hire people who committed that crime. Or imagine that,

in the meantime, the disabled man has withdrawn his request for a job. In both cases, despite X occurs, the first option does not hold and, all things considered, the second one can be applied in order to derogate to art. 3, 1st par., of the Italian constitution.

3. Defeasible Deontic Logic with strong permission

This section begins by introducing the language adopted to formalise obligations and strong permissions in DL, and describing the inferential mechanism in the form of proof conditions defining the logic. Finally, we show that the proposed formalisation enjoys properties appropriate to model the notion of strong permission. We will proceed incrementally: this section, as well as Section 4, works only with obligations and strong permissions expressed by rules for P. In Section 6 we will show how weak permissions and strong permissions based on defeaters can be easily captured in the framework.

We consider a logic whose language is defined as follows.

Definition 1 (Language). *Let PROP be a set of propositional atoms, $\text{MOD} = \{O, P\}$ the set of modal operators where O is the modality for the obligation and P for permission.*

- *The set $\text{Lit} = \text{PROP} \cup \{\neg p \mid p \in \text{PROP}\}$ denotes the set of literals.*
- *The complementary of a literal q is denoted by $\sim q$; if q is a positive literal p , then $\sim q$ is $\neg p$, and if q is a negative literal $\neg p$, then $\sim q$ is p .*
- *The set of modal literals is $\text{ModLit} = \{\Box l, \neg \Box l \mid l \in \text{Lit}, \Box \in \text{MOD}\}$.*

We introduce two preference operators, \otimes for obligations and \odot for permissions, and we will use \oslash when we refer to one of them generically. These operators are used to build chains of preferences, called \oslash -expressions. The formation rules for well-formed \oslash -expressions are:

- (a) every literal $l \in \text{Lit}$ is an \oslash -expression;
- (b) if A is an \otimes -expression, B is an \odot -expression and $c_1, \dots, c_k \in \text{Lit}$, then $A \otimes c_1 \otimes \dots \otimes c_k$ is an \otimes -expression, $B \odot c_1 \odot \dots \odot c_k$ is an \odot -expression, $A \odot B$ is an \oslash -expression;
- (c) every \otimes -expression and \odot -expression is an \oslash -expression;
- (d) nothing else is an \oslash -expression.

In addition we stipulate that \otimes and \odot obey the following properties:

1. $a \oslash (b \oslash c) = (a \oslash b) \oslash c$ (associativity);
2. $\oslash_{i=1}^n a_i = (\oslash_{i=1}^{k-1} a_i) \oslash (\oslash_{i=k}^n a_i)$ where there exists j such that $a_j = a_k$ and $j < k$ (duplication and contraction on the right).

Given an \oslash -expression A , the *length* of A is the number of literals in it. Given an \oslash -expression $A \oslash b \oslash C$ (where A and C can be empty), the *index* of b is the length of $A \oslash b$. We also say that b appears at index n in $A \oslash b$ if the length of $A \oslash b$ is n .

We adopt the standard Defeasible Logic definitions of *strict rules*, *defeasible rules*, and *defeaters* [5]. However for the sake of simplicity, and to better focus on the non-monotonic aspects that Defeasible Logic offers, in the remainder we use only defeasible rules and defeaters. In addition, we have to take the modal operators into account.

Definition 2 (Rules). *Let Lab be a set of arbitrary labels. Every rule is of the type*

$$r : A(r) \hookrightarrow C(r)$$

where

1. $r \in \text{Lab}$ is the name of the rule;
2. $A(r) = \{a_1, \dots, a_n\}$, the antecedent (or body) of the rule, is the set of the premises of the rule (alternatively, it can be understood as the conjunction of all the literals in it). Each a_i is either a literal, or a modal literal;
3. $\hookrightarrow \in \{\Rightarrow \Box, \leadsto\}$ denotes the type of the rule. If \hookrightarrow is $\Rightarrow \Box$, the rule is a defeasible rule, while if \hookrightarrow is \leadsto , the rule is a defeater. The subscript $\Box \in \text{MOD}$ in defeasible rules represents the modality introduced by the rule itself: the mode of a rule tells us what kind of conclusion we obtain from the rule. As we argued in Section 2, we do not need to label \leadsto with any modality;
4. $C(r)$ is the consequent (or head) of the rule, which is an \odot -expression. Two constraints apply on the consequent of a rule: (a) if \hookrightarrow is \leadsto , then $C(r)$ is a single literal; (b) if $\Box = \text{P}$, then $C(r)$ must be an \odot -expression.

Given a set of rules R , we will use the following abbreviations for specific subsets of rules:

- R_{def} denotes the set of all defeaters in the set R ;
- $R[q, n]$ is the set of rules where q appears at index n in the consequent. The set of (defeasible) rules where q appears at any index n is denoted by $R[q]$;
- R^\Box with $\Box \in \text{MOD}$ denotes the set of all rules in R introducing modality \Box ;
- $R^O[q, n]$ is the set of (defeasible) rules where q appears at index n and the operator preceding it is \otimes for $n > 1$ or the mode of the rule is O for $n = 1$. The set of (defeasible) rules where q appears at any index n satisfying the above constraints is denoted by $R^O[q]$;
- similarly $R^P[q, n]$ is the set of rules where q appears at index n , and the operator preceding it is \odot for $n > 1$ or the mode of the rule is P for $n = 1$. The set of (defeasible) rules where q appears at any index n satisfying the above constraints is denoted by $R^P[q]$.

Definition 3. A Defeasible Theory is a structure $D = (F, R, >)$, where F , the set of facts, is a set of literals and modal literals, R is a set of rules and $>$, the superiority relation, is a binary relation over R .

A theory corresponds to a normative system, i.e., a set of norms which are modelled by rules. The superiority relation is used for conflicting rules, i.e., rules whose conclusions are complementary literals, in case both rules fire. Notice that we do not impose any restriction on the superiority relation: it is just a binary relation determining the relative strength of two rules.

Definition 4. A proof P in a defeasible theory D is a linear sequence $P(1) \dots P(n)$ of tagged literals in the form of $+\partial_\Box q$ and $-\partial_\Box q$ with $\Box \in \text{MOD}$, where $P(1) \dots P(n)$ satisfy the proof conditions given in Definitions 8–11.

The tagged literal $+\partial_\Box q$ means that q is defeasibly provable in D with modality \Box , while $-\partial_\Box q$ means that q is defeasibly refuted with modality \Box . The initial part of length i of a proof P is denoted by $P(1..i)$.

The first thing to do is to define when a rule is applicable or discarded. A rule is *applicable* for a literal q if q occurs in the head of the rule, all non-modal literals in the antecedent are given as facts and all the modal literals have been defeasibly proved (with the appropriate modalities). On the other hand, a rule is *discarded* if at least one of the modal literals in the antecedent has not been proved (or is not a fact in the case

of non-modal literals). However, as literal q might not appear as the first element in an \odot -expression in the head of the rule, some additional conditions on the consequent of rules must be satisfied. Defining when a rule is applicable or discarded is essential to characterise the notion of provability for obligations ($\pm\partial_O$) and permissions ($\pm\partial_P$).

Definition 5. A rule $r \in R[q, j]$ such that $C(r) = c_1 \otimes \dots \otimes c_{l-1} \odot c_l \odot \dots \odot c_n$ is applicable for literal q at index j , with $1 \leq j < l$, in the condition for $\pm\partial_O$ iff

- (1) for all $a_i \in A(r)$:
 - (1.1) if $a_i = Ol$ then $+\partial_O l \in P(1..n)$;
 - (1.2) if $a_i = \neg Ol$ then $-\partial_O l \in P(1..n)$;
 - (1.3) if $a_i = Pl$ then $+\partial_P l \in P(1..n)$;
 - (1.4) if $a_i = \neg Pl$ then $-\partial_P l \in P(1..n)$;
 - (1.5) if $a_i = l \in \text{Lit}$ then $l \in F$, and
- (2) for all $c_k \in C(r)$, $1 \leq k < j$, $+\partial_O c_k \in P(1..n)$ and $(c_k \notin F \text{ or } \sim c_k \in F)$.

Conditions (1.1)–(1.5) represent the requirements on the antecedent as informally described above; condition (2) on the head of the rule states that each element c_k prior to q must be derived as an obligation, and a violation of such obligation has occurred.

Definition 6. A rule $r \in R[q, j]$ such that $C(r) = c_1 \otimes \dots \otimes c_{l-1} \odot c_l \odot \dots \odot c_n$ is applicable for literal q at index j , with $l \leq j \leq n$ in the condition for $\pm\partial_P$ iff

- (1) for all $a_i \in A(r)$:
 - (1.1) if $a_i = Ol$ then $+\partial_O l \in P(1..n)$;
 - (1.2) if $a_i = \neg Ol$ then $-\partial_O l \in P(1..n)$;
 - (1.3) if $a_i = Pl$ then $+\partial_P l \in P(1..n)$;
 - (1.4) if $a_i = \neg Pl$ then $-\partial_P l \in P(1..n)$;
 - (1.5) if $a_i = l \in \text{Lit}$ then $l \in F$, and
- (2) for all $c_k \in C(r)$, $1 \leq k < l$, $+\partial_O c_k \in P(1..n)$ and $(c_k \notin F \text{ or } \sim c_k \in F)$, and
- (3) for all $c_k \in C(r)$, $l \leq k < j$, $-\partial_P c_k \in P(1..n)$.

The only difference with respect to $\pm\partial_O$ is the presence of an additional condition, stating that all permissions prior to q must be refuted (condition (3)).

Definition 7. A rule $r \in R[q, j]$ such that $C(r) = c_1 \otimes \dots \otimes c_{l-1} \odot c_l \odot \dots \odot c_n$ is discarded for literal q at index j , with $1 \leq j \leq n$ in the condition for $\pm\partial_O$ or $\pm\partial_P$ iff

- (1) there exists $a_i \in A(r)$ such that
 - (1.1) if $a_i = Ol$ then $-\partial_O l \in P(1..n)$;
 - (1.2) if $a_i = \neg Ol$ then $+\partial_O l \in P(1..n)$;
 - (1.3) if $a_i = Pl$ then $-\partial_P l \in P(1..n)$;
 - (1.4) if $a_i = \neg Pl$ then $+\partial_P l \in P(1..n)$;
 - (1.5) if $a_i = l \in \text{Lit}$ then $l \notin F$, or
- (2) there exists $c_k \in C(r)$, $1 \leq k < l$, such that either $-\partial_O c_k \in P(1..n)$ or $c_k \in F$, or
- (3) there exists $c_k \in C(r)$, $l \leq k < j$, such that $+\partial_P c_k \in P(1..n)$.

In this case, condition (2) ensures that an obligation prior to q in the chain is not in force or has already been fulfilled (thus, no reparation is required), while condition (3) states that there exists at least one explicit derived permission prior to q .

We now introduce the proof conditions for $\pm\partial_O$ and $\pm\partial_P$.

Definition 8. *The proof condition of defeasible provability for obligation is*

$+ \partial_O$: If $P(n+1) = + \partial_O q$ then

(1) $Oq \in F$ or

(2.1) $O \sim q \notin F$ and $\neg Oq \notin F$ and $P \sim q \notin F$ and

(2.2) $\exists r \in R^O[q, i]$ such that r is applicable for q , and

(2.3) $\forall s \in R[\sim q, j]$, either

(2.3.1) s is discarded, or either

(2.3.2) $s \in R^O$ and $\exists t \in R[q, k]$ such that t is applicable for q and $t > s$, or

(2.3.3) $s \in R^P \cup R_{def}$ and $\exists t \in R^O[q, k]$ such that t is applicable for q and $t > s$.

To show that q is defeasibly provable as an obligation, there are two ways: (1) the obligation of q is a fact, or (2) q must be derived by the rules of the theory. In the second case, three conditions must hold: (2.1) q does not appear as not obligatory as a fact, and $\sim q$ is neither provable as an obligation nor as a permission using the set of modal facts at hand; (2.2) there must be a rule introducing the obligation for q which can apply; (2.3) every rule s for $\sim q$ is either discarded or defeated by a stronger rule for q . If s is an obligation rule, then it can be counterattacked by any type of rule; if s is a defeater or a permission rule, then only an obligation rule can counterattack it.

The strong negation of the above definition gives us the negative proof condition for obligation. Notice that the *strong negation* of a formula is closely related to the function that simplifies a formula by moving all negations to an inner most position in the resulting formula, and replaces the positive tags with the respective negative tags, and the other way around [6,17].

Definition 9. *The proof condition of defeasible refutability for obligation is*

$- \partial_O$: If $P(n+1) = - \partial_O q$ then

(1) $Oq \notin F$ and either

(2.1) $O \sim q \in F$ or $\neg Oq \in F$ or $P \sim q \in F$ or

(2.2) $\forall r \in R^O[q, i]$ either r is discarded for q , or

(2.3) $\exists s \in R[\sim q, j]$ such that

(2.3.1) s is applicable for $\sim q$, and

(2.3.2) if $s \in R^O$ then $\forall t \in R[q, k]$, either t is discarded or $t \not> s$, and

(2.3.3) if $s \in R^P \cup R_{def}$ then $\forall t \in R^O[q, k]$, either t is discarded or $t \not> s$.

We now introduce and briefly explain the proof conditions for permission.

Definition 10. *The proof condition of defeasible provability for permission is*

$+ \partial_P$: If $P(n+1) = + \partial_P q$ then

(1) $Pq \in F$ or

(2.1) $O \sim q \notin F$ and $\neg Pq \notin F$ and

(2.2) $\exists r \in R^P[q, i]$ such that r is applicable for q , and

(2.3) $\forall s \in R^O[\sim q, j]$, either

(2.3.1) s is discarded for $\sim q$, or

(2.3.2) $\exists t \in R[q, k]$ such that t is applicable for q and $t > s$.

This proof condition differs from its counterpart for obligation in two aspects: we allow scenarios where both $+ \partial_P q$ and $+ \partial_P \sim q$ hold, but $+ \partial_O \sim q$ must not hold (clause 2.1); any applicable rule s supporting $\sim q$ can be counterattacked by any type of rule t supporting

q , as s must be an obligation rule, and permission rules can only be attacked by obligation rules (clause 2.3).

As argued above, we define the negative proof condition for permission as the strong negation of that for $+\partial_P$.

Definition 11. *The proof condition of defeasible refutability for permission is*

- $-\partial_P$: If $P(n+1) = -\partial_P q$ then
- (1) $Pq \notin F$ and either
 - (2.1) $O\sim q \in F$ or $\neg Pq \in F$, or
 - (2.2) $\forall r \in R^P[q, i]$, either r is discarded, or
 - (2.3) $\exists s \in R^O[\sim q, j]$ such that
 - (2.3.1) s is applicable for $\sim q$, and
 - (2.3.2) $\forall t \in R[q, k]$, either t is discarded or $t \not\prec s$.

The logic resulting from the above proof conditions enjoys properties describing the appropriate behaviour of the modal operators.

Definition 12. *A Defeasible Theory $D = (F, R, >)$ is consistent iff $>$ is acyclic and F does not contain pairs of complementary (modal) literals, that is if D does not contain pairs like Ol and $\neg Ol$, Pl and $\neg Pl$, or l and $\sim l$. The theory D is O-consistent iff $>$ is acyclic and for any literal l , F does not contain any of the following pairs: Ol and $O\sim l$, Ol and $P\sim l$.*

As usual, given a Defeasible Theory D , we will use $D \vdash \pm \partial_\square l$ iff there is a proof P in D such that $P(n) = \pm \partial_\square l$ for an index n .

Proposition 13. *Let D be a consistent Defeasible Theory, and $\square \in \text{MOD}$. For any literal l , it is not possible to have both $D \vdash +\partial_\square l$ and $D \vdash -\partial_\square l$.*

Proof. It straightforwardly follows from the principle of strong negation proposed in [6,17]: indeed, the negative proof tags proposed in this work are defined as the strong negation of the positive ones. \square

The meaning of the above proposition is that it is not possible to prove that a literal is at the same time obligatory and not obligatory, or permitted and not permitted.

Proposition 14. *Let D be an O-consistent Defeasible Theory. For any literal l , it is not possible to have both $D \vdash +\partial_O l$ and $D \vdash +\partial_O \sim l$.*

Proof. We split the proof in two cases: (i) at least one of Ol and $O\sim l$ is in F , and (ii) none of them is in F .

For (i) the proposition immediately follows by the assumption of O-consistency. Suppose that $Ol \in F$. Then clause (1) of $+\partial_O$ holds for l . By O-consistency $O\sim l \notin F$, thus clause (1) of $+\partial_O$ does not hold for $\sim l$. Since $Ol \in F$, clause (2.1) of $+\partial_O$ is always falsified for $\sim l$, and the thesis is proved.

For (ii): First of all, it is easy to verify that no rule can be at the same time applicable and discarded for the derivation of $\pm \partial_O l(\sim l)$. Then, since both $+\partial_O l$ and $+\partial_O \sim l$ hold, we have that there are applicable obligation rules for both l and $\sim l$. This means that clause (2.3.2) holds for both $+\partial_O l$ and $+\partial_O \sim l$. Therefore, for every applicable rule for l there is an applicable rule for $\sim l$ stronger than the rule for l , and symmetrically, for every applicable rule for $\sim l$ there is an applicable rule for l stronger than the rule for $\sim l$. Since the set of rules in a theory is finite, the situation we have just described is possible only

if there is a cycle in the transitive closure of the superiority relation. Therefore, we have a contradiction because the superiority relation is assumed to be acyclic (the transitive closure of the relation does not contain cycles). \square

The meaning of the proposition is that no formula is both obligatory and forbidden at the same time. However, the proposition does not hold for permission. It is possible to have both the explicit permission of l and the explicit permission of $\sim l$.

The relationships between permissions and obligations are governed by the following proposition:

Proposition 15. *Let D be an O-consistent Defeasible Theory. For any literal l :*

1. *if $D \vdash +\partial_O l$, then $D \vdash -\partial_O \sim l$;*
2. *if $D \vdash +\partial_O l$, then $D \vdash -\partial_P \sim l$;*
3. *if $D \vdash +\partial_P l$, then $D \vdash -\partial_O \sim l$.*

Proof. 1. Let D be an O-consistent Defeasible Theory, and $D \vdash +\partial_O l$. Literal $\sim l$ can be in only one of the following mutually exclusive situations: (i) $D \vdash +\partial_O \sim l$; (ii) $D \vdash -\partial_O \sim l$; (iii) $D \not\vdash \pm \partial_O \sim l$. Proposition 14 allows us to exclude case (i) since $D \vdash +\partial_O l$ by hypothesis. Situation (iii) denotes situations where there are loops in the theory involving literal $\sim l$ ⁴, but inevitably this would affect also the provability of literal l , i.e., we would not be able to give a proof for $+\partial_O l$ as well. This is in contradiction with the hypothesis; thus, situation (ii) must be the case.

2. Let D be an O-consistent Defeasible Theory, and $D \vdash +\partial_O l$. By definition of proof conditions for $+\partial_O$, statements (1)–(2.3.3) hold.

If $Ol \in F$, then condition (2.1) of $-\partial_P$ holds for $\sim l$. Furthermore, by O-consistency of D , $Ol \in F$ implies condition (1) of $-\partial_P$ for $\sim l$, and we obtain the thesis.

Otherwise, from condition (2.2) of $+\partial_O$, conditions (2.3) and (2.3.1) of $-\partial_P$ follow. Again, we can iterate the same reasoning for condition (2.3.1) of $+\partial_O$ implying condition (2.2) of $-\partial_P$. It remains to consider conditions (2.3.2) and (2.3.3) of $+\partial_O$. In the first case, the attacking rule s is an obligation rule, thus it is of no interest in this proof since in condition (2.2) of $-\partial_P$ we consider only permission rules. Thus, for the latter case, we know there exists a rule t for obligation that is stronger than s , and this tuple of rules (t, s) in condition (2.3.3) for $+\partial_O$ is the equivalent but opposite tuple of the rules used in condition (2.3.2) for $-\partial_P$. But, to analyse in an exhaustive way condition (2.3.2) of $-\partial_P$, we have to take into consideration the whole set of rules $S = \{s_1, \dots, s_n\}$ for permission leading to $\sim l$, against the set of rules $T = \{t_1, \dots, t_m\}$ for obligation leading to l .⁵ For each subset S' of S , every rule $s' \in S'$ is either discarded, or there exists a rule $t' \in T$ that is stronger (the existence of t' is guaranteed by the fact that $+\partial_O l$ holds by hypothesis). We can now remove S' from S (as it cannot be used for proving $+\partial_P \sim l$), reducing the number of elements in S . The iteration of this procedure eventually empties the set S since the number of rules in D is finite, and the superiority relation is acyclic.

3. Let D be an O-consistent Defeasible Theory, and $D \vdash +\partial_P l$. By definition of proof conditions for $+\partial_P$, statements (1) or (2.1)–(2.3.2) hold. The proof follows step by step that of Part 2 of the proposition. Moreover, steps for conditions (1)–(2.2) are the mere

⁴For examples situations like $O \sim l \Rightarrow_O \sim l$, where the proof conditions will generate a loop without introducing a proof.

⁵Notice that the rules in conditions (2.2) and (2.3.3) are different rules: they form a team that can defeat teams of rules for the opposite.

juxtaposition. It remains to analyse the interrelationship between condition (2.3) of $+\partial_P$ and conditions (2.3.2)–(2.3.3) of $-\partial_O$. Since condition (2.3) of $+\partial_P$ takes into account only rules for obligation which are systematically defeated with an analogous process of rule elimination, thus conditions (2.3.2) and (2.3.3) of $-\partial_O$ are satisfied. \square

The combination of Part 2 and 3 of Proposition 15 describes the consistency between obligation and permission. Part 3 also gives the relationships between strong and weak permission. As we discussed in Section 1, a weak permission is a permission obtained from the failure to derive the opposite obligation. This means that we have the weak permission of p when we have $-\partial_O \sim p$, and Part 3 guarantees that we have it when $+\partial_P p$ holds.

We conclude this section showing how the logic developed hitherto works with the example introduced at the end of Section 2.

Example. Let us recall the scenario reported at the end of Section 2, and formally explain the conclusions of the theory using applicability of rules and proof tags as defined above. The primary obligation to call the ambulance is obtained (i.e., we derive $+\partial_O \text{CallAmbulance}$), but the obligation is violated as $\neg \text{CallAmbulance} \in F$, making r_1 applicable for *Help*; rule r_3 is applicable for literal $\neg \text{Help}$ and could attack r_1 , but $r_1 > r_3$, thus we have also $+\partial_O \text{Help}$. Also CallFiremen is derived as an obligation but it is violated, thus rule r_2 is applicable for literal *Extinguish* in the condition for $+\partial_O$. As $+\partial_O \text{Help}$ holds, r_3 is applicable for $\neg \text{Extinguish}$, and $r_3 > r_2$, thus we derive $+\partial_P \neg \text{Extinguish}$.

4. Algorithms for defeasible extension

We now present procedures and algorithms apt to compute the *extension* of a *finite* Defeasible Theory, i.e., a theory where the set of facts and rules is finite, in order to bound the complexity of the logic introduced in the previous sections. The algorithms are based on the algorithm proposed by Maher [23] to show that Defeasible Logic has linear complexity; the algorithms also incorporate the notion of inferiorly defeated rules proposed by [22] to handle directly the superiority relation.

This section is divided in three main parts. In the first part we give the formal definitions and introduce the notation adopted. The second part, which contains the main body, describes the required computations: Algorithms 5 and 4 effectively compute the defeasible extension of a Defeasible Theory given as an input, while Algorithms 1, 2, and 3 are but auxiliary procedures that execute all the necessary operations due to any modification of the extension. Each algorithm is followed by a technical explanation on how it works. In the third part we present formal properties that are meant to prove the computational results proposed in Section 5.

4.1. Notation for the algorithms

We introduce the notation relevant to our framework.

Given a Defeasible Theory D , HB_D is the set of literals such that the literal or its complement appears in D , where ‘appears’ means that it is a sub-formula of a modal literal occurring in the theory. The modal Herbrand Base of D is $HB = \{\Box l \mid \Box \in \text{MOD}, l \in HB_D\}$. Accordingly, the extension of a Defeasible Theory is defined as follows.

Definition 16. Given a Defeasible Theory D , the defeasible extension of D is defined as

$$E(D) = (+\partial_O, +\partial_P, -\partial_O, -\partial_P),$$

where $\pm\partial_\square = \{l \in HB_D : D \vdash \pm\partial_\square l\}$ with $\square \in \text{MOD}$. We define two Defeasible Theories D and D' to be equivalent (in notation $D \equiv D'$) if they have the same extensions, i.e., $E(D) = E(D')$.

The next definition introduces two syntactical operations on the consequent of rules used by the algorithms, whose meaning will be clear in the remainder.

Definition 17. Let $c_1 = a_1 \odot \dots \odot a_{l-1}$ and $c_2 = a_{i+1} \odot \dots \odot a_n$ be two (possibly empty) \odot -expressions such that a_i does not occur in them, and $c = c_1 \odot a_i \odot c_2$ is an \odot -expression. Let r be a rule with form $A(r) \hookrightarrow c$. We define the operation of truncation of the consequent c at a_i as:

$$A(r) \hookrightarrow c!a_i = A(r) \hookrightarrow c_1 \odot a_i.$$

We define the removal of a_i from the consequent c , $A(r) \hookrightarrow c \ominus a_i$, as:

$$A(r) \hookrightarrow c \ominus a_i = \begin{cases} A(r) \Rightarrow_O c_1 \otimes c_2 & \text{if } r \text{ is } A(r) \Rightarrow_O c_1 \otimes a_i \otimes c_2 \\ A(r) \Rightarrow_O c_1 \odot c_2 & \text{if } r \text{ is } A(r) \Rightarrow_O c_1 \otimes a_i \odot c_2 \\ A(r) \Rightarrow_{\square \in \text{MOD}} c_1 \odot c_2 & \text{if } r \text{ is } A(r) \Rightarrow_{\square \in \text{MOD}} c_1 \odot a_i \odot c_2 \\ A(r) \Rightarrow_P c_2 & \text{if } r \text{ is } A(r) \Rightarrow_O a_i \odot c_2 \end{cases}$$

The next definition extends the concept of complement presented in Section 3 for modal literals, and it is used to establish the logical connection among proved and refuted literals in our framework.

Definition 18. We define the complement of a given literal l , denoted by \tilde{l} , as:

1. If $l \in \text{Lit}$, then $\tilde{l} = \{\sim l\}$;
2. If $l = Om$, then $\tilde{l} = \{\neg Om, O\sim m, P\sim m\}$;
3. If $l = \neg Om$, then $\tilde{l} = \{Om\}$;
4. If $l = Pm$, then $\tilde{l} = \{\neg Pm, O\sim m\}$;
5. If $l = \neg Pm$, then $\tilde{l} = \{Pm\}$.

Given $\square \in \text{MOD}$, the sets $\pm\partial_\square$ denote the global sets of defeasible conclusions (i.e., the set of literals for which condition $\pm\partial_\square$ holds), while ∂_\square^\pm are the corresponding temporary sets. Notice that the complement of $\neg Pm$ does not include Om (and vice versa) because the failure to derive Pm cannot depend on the derivation of Om , but rather on the fact that $O\sim m$ is the case.

4.2. Algorithms

We begin this subsection by reporting and explaining the three auxiliary procedures used in the two main algorithms for the computation of the extension of a logic.

Algorithm 1 DISCARD performs all operations related to when $-\partial_\square l$ holds for a given literal l .

First of all, literal l is placed in the local set of refuted literals with modality \square (line 2). Furthermore, condition $-\partial_\square l$ makes literal $\neg\square l$ provable, therefore it can be safely removed from all rules where it appears as an antecedent, being that its contribution to a rule being applicable or refuted has already been established. Since l cannot be

Algorithm 1 Discard

```
1: procedure DISCARD( $l \in \text{Lit}, \Box \in \{P, O\}$ )
2:    $\partial_{\Box}^- \leftarrow \partial_{\Box}^- \cup \{l\}$ 
3:    $R \leftarrow \{A(r) \setminus \{\neg \Box l\} \hookrightarrow C(r) \mid r \in R\} \setminus \{r \in R \mid \Box l \in A(r)\}$ 
4:    $>\leftarrow> \setminus \{(r, s), (s, r) \in > \mid \Box l \in A(r)\}$ 
5:    $HB \leftarrow HB \setminus \{\Box l\}$ 
6: end procedure
```

proved with modality \Box , every rule containing $\Box l$ in its body is discarded by clauses (1.1) and (1.3) of Definition 7, and thus we can remove such rules without affecting the conclusions that can be derived from the theory (line 3). In addition, we remove all pairs involving the rules from the superiority relation (line 4), and $\Box l$ from the modal Herbrand Base (line 5).

Algorithms 2 MODIFYOBL and 3 MODIFYPERM behave in a very similar way: both of them modify the theory to accommodate the positive derivation of a modal literal. They only differ on the kind of rules they manipulate (obligation and permission rules, respectively).

Algorithm 2 ModifyObl

```
1: procedure MODIFYOBL( $l \in \text{Lit}$ )
2:    $\partial_O^+ \leftarrow \partial_O^+ \cup \{l\}$ 
3:    $\partial_O^- \leftarrow \partial_O^- \cup \{\sim l\}$ 
4:    $\partial_P^- \leftarrow \partial_P^- \cup \{\sim l\}$ 
5:    $HB \leftarrow HB \setminus \{Ol, O\sim l, P\sim l\}$ 
6:   if  $Ol \notin F$  then
7:      $R \leftarrow \{A(r) \setminus \{Ol, \neg O\sim l\} \hookrightarrow C(r) \mid r \in R\} \setminus \{r \in R \mid A(r) \cap \widetilde{Ol} \neq \emptyset\}$ 
8:      $>\leftarrow> \setminus \{(r, s), (s, r) \in > \mid A(r) \cap \widetilde{Ol} \neq \emptyset\}$ 
9:   end if
10:   $R \leftarrow \{A(r) \hookrightarrow C(r) \ominus l \mid r \in R^O[l, n] \text{ for an index } n\}$ 
11:   $R \leftarrow \{A(r) \hookrightarrow C(r) \ominus \sim l \mid r \in R^P[\sim l, n] \text{ for an index } n\}$ 
12:   $R \leftarrow \{A(r) \hookrightarrow C(r)! \sim l \ominus \sim l \mid r \in R^O[\sim l, n] \text{ for an index } n\}$ 
13: end procedure
```

The input of both procedures is a literal l . As such, we add it to the corresponding set of derived literals (line 2). Since $D \vdash +\partial_O l$ implies $D \vdash -\partial_O \sim l, -\partial_P \sim l$ by Proposition 15 Part 1 and 2, and $D \vdash +\partial_P l$ implies $D \vdash -\partial_O \sim l$ by Proposition 15 Part 3, we also remove $\sim l$ from the appropriate sets of refuted literals; then the modal literal along with the set of its complementaries⁶ are removed from the Modal Herbrand Base (lines 3–5 and 3–4, respectively). Lines 6–9 and 5–8, respectively, follow the same reasoning of line 3 in Algorithm 1 DISCARD. Finally, the rules of the theory are modified on account of the modality the literal is derived with as well as the conditions for the applicability of a rule given in Definitions 5–7 (lines 10–12 and 9–10, respectively).

⁶Notice that we do not remove any negative modal literal from HB by definition of modal Herbrand Base.

Algorithm 3 ModifyPerm

```
1: procedure MODIFYPERM( $l \in \text{Lit}$ )
2:    $\partial_P^+ \leftarrow \partial_P^+ \cup \{l\}$ 
3:    $\partial_O^- \leftarrow \partial_O^- \cup \{\sim l\}$ 
4:    $HB \leftarrow HB \setminus \{Pl, O\sim l\}$ 
5:   if  $Pl \notin F$  then
6:      $R \leftarrow \{A(r) \setminus \{Pl, \neg O\sim l\} \hookrightarrow C(r) \mid r \in R\} \setminus \{r \in R \mid A(r) \cap \tilde{Pl} \neq \emptyset\}$ 
7:      $\succ \leftarrow \succ \setminus \{(r, s), (s, r) \in \succ \mid A(r) \cap \tilde{Pl} \neq \emptyset\}$ 
8:   end if
9:    $R \leftarrow \{A(r) \hookrightarrow C(r)! \sim l \ominus \sim l \mid r \in R^O[\sim l, n] \text{ for an index } n\}$ 
10:   $R \leftarrow \{A(r) \hookrightarrow C(r)! l \mid r \in R^P[l, n] \text{ for an index } n\}$ 
11: end procedure
```

Algorithm 4 CheckFacts

```
1: procedure CHECKFACTS
2:   for  $l \in F$  do
3:      $R \leftarrow \{A(r) \setminus \{l\} \hookrightarrow C(r) \mid r \in R\} \setminus \{r \in R \mid A(r) \cap \tilde{l} \neq \emptyset\}$ 
4:      $\succ \leftarrow \succ \setminus \{(r, s), (s, r) \in \succ \mid A(r) \cap \tilde{l} \neq \emptyset\}$ 
5:     if  $l \in \text{Lit}$  then
6:        $R \leftarrow \{A(r) \hookrightarrow C(r)! l \mid r \in R^O[l, n] \text{ for an index } n\}$ 
7:     end if
8:     if  $l = Om$  then
9:       MODIFYOBL( $m$ )
10:    end if
11:    if  $l = \neg Om$  then
12:       $\neg \partial_O \leftarrow \neg \partial_O \cup \{m\}$ 
13:       $HB \leftarrow HB \setminus \{Om\}$ 
14:       $R \leftarrow \{A(r) \hookrightarrow C(r)! m \ominus m \mid r \in R^O[m, n] \text{ for an index } n\}$ 
15:    end if
16:    if  $l = Pm$  then
17:      MODIFYPERM( $m$ )
18:    end if
19:    if  $l = \neg Pm$  then
20:       $\neg \partial_P \leftarrow \neg \partial_P \cup \{m\}$ 
21:       $HB \leftarrow HB \setminus \{Pm, Om\}$ 
22:       $R \leftarrow \{A(r) \setminus \hookrightarrow C(r) \ominus m \mid r \in R^P[m, n] \text{ for an index } n\}$ 
23:    end if
24:  end for
25: end procedure
```

Before describing how Algorithm 4 works, let us recall some concepts about the provability of a literal. Given a Defeasible Theory, a modal literal $\Box l \in F$ is trivially proved with the corresponding modality by definition. Furthermore, we also stated that a non-modal literal is proved within the theory if it is a fact.

Based on these facts, the procedure described in Algorithm 4 CHECKFACTS begins by removing all factual literals from every rule where they appear as an antecedent; it also

removes all rules whose body contains a complementary literal (line 3). The superiority relation is then modified in view of this operation (line 4).

From this point on, different operations are performed on account of which kind of factual literal is considered.

1. If l is a non-modal literal, we truncate the head of all rules at l , where l appears as an obligation (lines 5–7);
2. if l is a positive modal literal for obligation (lines 8–10) or permission (lines 16–18), then Algorithm 2 MODIFYOBL (respectively Algorithm 3 MODIFYPERM) is called to properly modify the theory. Notice that operations in lines 7–8 of Algorithm 2 MODIFYOBL and 6–7 of Algorithm 3 MODIFYPERM are not performed in this case, since they are equivalent to lines 3–4 of Algorithm 4 CHECKFACTS;
3. if l is a negative modal literal for obligation $\neg Om$ (lines 11–15), then $-\partial_O m$ holds, and clause (2) of Definition 7 makes all rules containing m as an obligation in their heads discarded for all literals after m . Hence, we truncate all these chains at m , and then remove m (line 14);
4. if l is a negative modal literal for permission $\neg Pm$ (lines 19–23), then $-\partial_P m$ holds in the theory. Thus, we remove m in every chain where m appears as a permission (line 22).

We conclude this section by presenting and describing Algorithm 5 COMPUTEDEFEASIBLE, which represents the main core for the computation of the defeasible extension of a theory.

In lines 1–5, we initialise variables $+\partial_O$, $+\partial_P$ and, for each literal l , a set $R[l]_{inf d}$ containing all the rules for l that are defeated by a rule for the opposite. Algorithm 4 CHECKFACTS is invoked to compute all defeasible conclusions derived from the set of facts (line 6).

The algorithm consists of a main loop (the **repeat** cycle in lines 7–35) that performs a series of transformations to reduce a Defeasible Theory into a simpler equivalent one. The loop ends when no more modifications on the extension are made, i.e., when both variables ∂_O^+ and ∂_O^- are empty at the end of an iteration.

At the beginning of the cycle, we re-initialise the set of conclusions computed at the iteration of the main loop (lines 8–9).

The **for** cycle in lines 10–32 checks all the rules for every literal l of the theory. In lines 11–13 it modifies the theory invoking Algorithm 1 DISCARD for all modal literals with no supporting chains. Lines 14–31 loop over all rules in the theory for the current literal l , and checks if an applicable rule exists with l as first element in its head. If the rule introduces l as an obligation (lines 14–22), then we have to collect all rules for the opposite, and check if they are all defeated by a rule for l . If this is the case, then we have proved $+\partial_O l$ and Algorithm 2 MODIFYOBL must be invoked.

On the other hand, if l is introduced as a permission (lines 23–31), then we have to take into account only obligation rules for $\sim l$, and to check if every rule for $\sim l$ as an obligation is defeated by at least one rule for the opposite. If so, condition $+\partial_P l$ holds, and Algorithm 3 MODIFYPERM is invoked. Finally, all modifications on the extension, due to the execution of the cycle, are stored in the global sets of conclusions (lines 33–34).

Algorithm 5 ComputeDefeasible

Input: A defeasible theory D .

Output: The extension $E(D)$ of D .

```
1: for  $\square \in \{O, P\}$  do
2:    $+\partial_{\square} \leftarrow \emptyset$ 
3:    $-\partial_{\square} \leftarrow \emptyset$ 
4: end for
5:  $R[l]_{inf} \leftarrow \emptyset$  for each  $l \in \text{Lit}$ 
6: CHECKFACTS
7: repeat
8:    $\partial_{\square}^+ \leftarrow \emptyset$ 
9:    $\partial_{\square}^- \leftarrow \emptyset$ 
10:  for  $\square l \in HB, \square \in \{O, P\}$  do
11:    if  $R^{\square}[l] = \emptyset$  then
12:      DISCARD( $l, \square$ )
13:    end if
14:    if there exists  $r \in R^O[l, 1]$  such that  $A(r) = \emptyset$  then
15:       $R[\sim l]_{inf} \leftarrow R[\sim l]_{inf} \cup \{s \in R[\sim l] \mid r > s\}$ 
16:      if  $\{s \in R[\sim l] \mid s > r\} = \emptyset$  then
17:        DISCARD( $\sim l, \square$ )
18:        if  $R[\sim l] \setminus R[\sim l]_{inf} = \emptyset$  and  $\neg Ol \notin F$  then
19:          MODIFYOBL( $l$ )
20:        end if
21:      end if
22:    end if
23:    if there exists  $r \in R^P[l, 1]$  such that  $A(r) = \emptyset$  then
24:       $R[\sim l]_{inf} \leftarrow R[\sim l]_{inf} \cup \{s \in R^O[\sim l] \mid r > s\}$ 
25:      if  $\{s \in R[\sim l] \mid s > r\} = \emptyset$  then
26:        DISCARD( $\sim l, O$ )
27:        if  $R^O[\sim l] \setminus R[\sim l]_{inf} = \emptyset$  then
28:          MODIFYPERM( $l$ )
29:        end if
30:      end if
31:    end if
32:  end for
33:   $+\partial_{\square} \leftarrow +\partial_{\square} \cup \partial_{\square}^+$ 
34:   $-\partial_{\square} \leftarrow -\partial_{\square} \cup \partial_{\square}^-$ 
35: until  $\partial_{\square}^+ = \emptyset$  and  $\partial_{\square}^- = \emptyset$ 
36: return  $E(D) = (+\partial_O, +\partial_P, -\partial_O, -\partial_P)$ 
```

4.3. Properties of defeasible theory transformations

The properties we are going to show below are related to operations that transform a theory D into an equivalent simpler theory D' (where by the term ‘simpler’ we mean a theory with a minor number of symbols in it).

The transformations operate either by removing some elements from it, or by deleting a rule from the theory. Given the functional nature of the operations, we will refer to

the rules in the target theory with the same names/labels as the rules in the source theory. Thus, given a rule $r \in D$, we will refer to the rule corresponding to it in D' (if it exists) with the same label, namely r .

For the sake of readability, the proofs of all the theoretical results (Lemmas 19–28) are not reported in this subsection and the interested reader can find them in Appendix A.

Given a non-modal literal $p \in F$, we can obtain an equivalent theory by removing p in every rule where it appears in the antecedent. Moreover, if the rule is for an obligation, Definition 7 clause (2) ensures that the rule will be discarded for every element after p , and therefore we can truncate the reparation chain at p . Instead, if the rule is for a permission, we cannot operate on it. In both cases, we only consider rules where the complement of p does not appear in the antecedent. Finally, the superiority relation can be simplified by removing all tuples with a rule containing $\sim p$ in the antecedent, or an obligation rule for an element after p in its consequent.

Lemma 19. *Let $D = (F, R, >)$ be a theory such that $p \in F \cap \text{Lit}$. Let $D' = (F', R', >')$ be the theory obtained from D where*

$$\begin{aligned} F' &= F \setminus \{p\} \\ R' &= \{r : A(r) \setminus \{p\} \Rightarrow_O C(r)!p \mid r \in R, A(r) \cap \tilde{p} = \emptyset\} \cup \\ &\quad \{r : A(r) \setminus \{p\} \Rightarrow_P C(r) \mid r \in R, A(r) \cap \tilde{p} = \emptyset\} \\ >' &= > \setminus \{(r, s), (s, r) \mid r, s \in R, A(r) \cap \tilde{p} \neq \emptyset\}. \end{aligned}$$

Then $D \equiv D'$.

Starting from the modified theory given by the transformations of the previous lemma, we now consider a theory with only modal literals in the set of facts. If a literal p is provable as an obligation, then we can simplify the theory by removing Op in every antecedent of the rules in R , and erase the rules where at least one element of \widetilde{Op} appears in the antecedent. Since by hypothesis $F \cap \text{Lit} = \emptyset$, if p is present in the reparation chain of an obligation rule, we simplify the theory by removing p from the consequent. If $\sim p$ is in the consequent, we can also truncate the reparation chain of the rule since, by Definition 7 clause (2), the rule will be discarded for each element after $\sim p$ (Proposition 15 Part 1 states that $-\partial_O \sim p$ holds as well). Moreover, Proposition 15 Part 2 ensures that $-\partial_P \sim p$ holds. Thus, Definition 6 clause (3) allows us to remove $\sim p$ in the consequent of permission rules for $\sim p$. Finally, the superiority relation can be simplified by removing all tuples with a rule containing at least one element of \widetilde{Op} in the antecedent.

Lemma 20. *Let $D = (F, R, >)$ be a theory such that $F \cap \text{Lit} = \emptyset$ and $D \vdash +\partial_O p$. Let $D' = (F, R', >')$ be the theory obtained from D where*

$$\begin{aligned} R' &= \{r : A(r) \setminus \{Op\} \Rightarrow_O C(r)! \sim p \ominus \sim p \mid r \in R, A(r) \cap \widetilde{Op} = \emptyset\} \cup \\ &\quad \{r : A(r) \setminus \{Op\} \Rightarrow_O C(r) \ominus p \mid r \in R, A(r) \cap \widetilde{Op} = \emptyset\} \cup \\ &\quad \{r : A(r) \setminus \{Op\} \Rightarrow_P C(r) \ominus \sim p \mid r \in R, A(r) \cap \widetilde{Op} = \emptyset\} \\ >' &= > \setminus \{(r, s), (s, r) \mid r, s \in R, A(r) \cap \widetilde{Op} \neq \emptyset\}. \end{aligned}$$

Then $D \equiv D'$.

As the previous lemma, we consider a theory with only modal literals in the set of facts. Since the theory proves $-\partial_O p$, also $\neg Op$ holds. Thus, we obtain an equivalent simpler theory by erasing all rules with Op as one of the antecedents, and by removing $\neg Op$ in each rule where it appears in the antecedent. Again, by Definition 7 clause (2), for every obligation rule we can truncate each reparation chain with p in the consequent and eliminate it from such a chain. Finally, the superiority relation can be simplified by removing all the pairs with a rule containing Op in the antecedent.

Lemma 21. *Let $D = (F, R, >)$ be a theory such that $F \cap \text{Lit} = \emptyset$ and $D \vdash -\partial_O p$. Let $D' = (F, R', >')$ be theory obtained from D where*

$$\begin{aligned} R' &= \{r : A(r) \setminus \{\neg Op\} \Rightarrow_{\square} C(r) \mid r \in R, A(r) \cap \{Op\} = \emptyset\} \cup \\ &\quad \{r : A(r) \Rightarrow_O C(r)!p \ominus p \mid r \in R, A(r) \cap \{Op\} = \emptyset\} \\ >' &= > \setminus \{(r, s), (s, r) \mid r, s \in R, A(r) \cap \{Op\} \neq \emptyset\}. \end{aligned}$$

Then $D \equiv D'$.

We can defeasibly prove a literal p as a permission. A simpler equivalent theory is one where we remove Pp in each set of antecedents and where we erase all the rules containing at least one element of the complement of \widetilde{Pp} in the antecedent. Proposition 15 Part 3 states that $-\partial_O \sim p$ holds. Thus, by Definition 7 clause (2), if $\sim p$ appears in the reparation chain of an obligation rule, we can remove it after having truncate the chain at $\sim p$. Instead, if we consider permission rules with p in the consequent, by Definition 7 clause (3), we can truncate the corresponding reparation chain at p . Finally, the superiority relation can be simplified by removing all the pairs with a rule with an element of Pp in the antecedent.

Lemma 22. *Let $D = (F, R, >)$ be a theory such that $F \cap \text{Lit} = \emptyset$ and $D \vdash +\partial_P p$. Let $D' = (F, R', >')$ be theory obtained from D where*

$$\begin{aligned} R' &= \{r : A(r) \setminus \{Pp\} \Rightarrow_O C(r)! \sim p \ominus \sim p \mid r \in R, A(r) \cap \widetilde{Pp} = \emptyset\} \cup \\ &\quad \{r : A(r) \setminus \{Pp\} \Rightarrow_P C(r)!p \mid r \in R, A(r) \cap \widetilde{Pp} = \emptyset\} \\ >' &= > \setminus \{(r, s), (s, r) \mid r, s \in R, A(r) \cap \widetilde{Pp} \neq \emptyset\}. \end{aligned}$$

Then $D \equiv D'$.

The theory proves $-\partial_P p$, allowing $\neg Pp$ to hold. Thus, we obtain an equivalent theory if we erase all the rules with Pp in the set of the antecedents and if we remove $\neg Pp$ where it appears in the tail of a rule. Moreover, if the rule is for a permission, we remove p from the reparation chain. Finally, we change the superiority relation by erasing the tuples with a rule with Pp in the antecedent.

Lemma 23. *Let $D = (F, R, >)$ be a theory such that $F \cap \text{Lit} = \emptyset$ and $D \vdash -\partial_P p$. Let $D' = (F, R', >')$ be theory obtained from D where*

$$\begin{aligned} R' &= \{r : A(r) \setminus \{\neg Pp\} \Rightarrow_{\square} C(r) \mid r \in R, A(r) \cap \{Pp\} = \emptyset\} \cup \\ &\quad \{r : A(r) \Rightarrow_P C(r) \ominus p \mid r \in R, A(r) \cap \{Pp\} = \emptyset\} \\ >' &= > \setminus \{(r, s), (s, r) \mid r, s \in R, A(r) \cap \{Pp\} \neq \emptyset\}. \end{aligned}$$

Then $D \equiv D'$.

The following two lemmas represent conditions under which a literal can be proved as an obligation or as a permission. The transformations dictated by the previous lemmas empty the antecedent of every applicable rule.

Definition 24. Given a theory $D = (F, R, >)$, and a set of rules S , the subset of S of inferiorly defeated rules for a literal p , $S[p]_{inf}$ is thus defined: $r \in S[p]_{inf}$ iff

1. $\exists s \in R[\sim p]$ such that $A(r) = \emptyset$ and $s > r$, and
2. if $r \in R^P[p]$, then $s \in R^O[\sim p]$.

Lemma 25. Let $D = (F, R, >)$ be a theory such that $F \cap \text{ModLit} = \emptyset$, $\exists r \in R^O[p, 1]$, $A(r) = \emptyset$, and $R[\sim p] \subseteq R_{inf}$. Then $D \vdash +\partial_O p$.

Lemma 26. Let $D = (F, R, >)$ be a theory such that $F \cap \text{ModLit} = \emptyset$, $\exists r \in R^P[p, 1]$, $A(r) = \emptyset$, and $R^O[\sim p] \subseteq R_{inf}$. Then $D \vdash +\partial_P p$.

The next two lemmas concern conditions to determine when it is possible to assert that a literal is negatively provable.

Lemma 27. Let $D = (F, R, >)$ be a theory such that $F \cap \text{ModLit} = \emptyset$ and $R^\square[p] = \emptyset$, for $\square \in \{O, P\}$. Then $D \vdash -\partial_\square p$.

Lemma 28. Let $D = (F, R, >)$ be a theory such that $F \cap \text{ModLit} = \emptyset$, and $\exists r \in R[p, 1]$ such that $A(r) = \emptyset$ and $r_{sup} = \emptyset$. Then

1. if $r \in R^O$, then $D \vdash -\partial_\square \sim p$, $\square \in \{O, P\}$;
2. if $r \in R^P \cup R_{def}$, then $D \vdash -\partial_O \sim p$.

5. Computational results

In this section we present the computational properties of the algorithms previously described. Since, as stated above, the first three algorithms are sub-routines of the two main ones, we will present the correctness and completeness results of these algorithms inside theorems for Algorithms 4 CHECKFACTS and 5 COMPUTEDEFEASIBLE.

In order to properly exhibit results on the complexity of the algorithms, we need the following definition.

Definition 29. Given a Defeasible Theory D , the size S of D is the number of literal occurrences plus the number of the rules in D .

We also report some key ideas and intuitions behind our implementation.

1. Each operation on global sets $\pm\partial_\square$ and ∂_\square^\pm requires a constant time, as we manipulate finite sets of literals;
2. For each literal $\square l \in HB$, we implement a hash table with pointers to rules where the literal occurs; thus, retrieving the set of rules containing a given literal requires constant time.
3. The superiority relation can also be implemented by means of hash tables; once again, the information required to modify a given tuple can be accessed in constant time.

Theorem 30. Given a modal Defeasible Theory D with size S , Algorithm 4 CHECKFACTS terminates and its computational complexity is $O(S)$.

Proof. Termination of Algorithm 4 CHECKFACTS is given by definition of modal De-feasible Theory, since the internal sub-routines (i.e., Algorithm 2 MODIFYOBL and 3 MODIFYPERM), as well as the algorithm itself, manipulate finite sets of rules and facts.

For a correct analysis of the complexity of Algorithm 4 CHECKFACTS, it is of the utmost importance to correctly comprehend Definition 29. Here we underline that the size S of a theory represents the total number of occurrences of each literal in every rule of such a theory. Let us examine a theory with Y literals and whose size is Z . If we consider a cycle whose purpose is to call, for each literal, a procedure that selectively deletes it from all the rules of the theory (no matter to what end), a rough computational complexity would be $O(Z^2)$. In fact, the complexity of the procedure by itself is bound to the number of rules in the theory, which is in $O(Z)$, and this procedure is iterated $Y \in O(Z)$ times.

However, a more fined-grained analysis shows that the complexity of this loop is lower. The mistake of the previous analysis is that it considers the complexity of the procedure separately from the complexity of the external loop, whilst they are strictly dependent. Indeed, the overall number of operations made by the sum of all loop iterations cannot outrun the number of occurrences of the literals, $O(Z)$, because the operations in the inner procedure directly decrease, iteration after iteration, the number of the remaining repetitions of the outmost loop, and the other way around. Therefore, the overall complexity is not bound to $O(Z) \cdot O(Z) = O(Z^2)$, but to $O(Z) + O(Z) = O(Z)$.

We can contextualise the above reasoning to Algorithm 4 CHECKFACTS. The main cycle in lines 2–26 is iterated over the set of facts, whose cardinality is in $O(S)$; the operations in lines 10 and 19 (invoking Algorithms 2 MODIFYOBL and 3 MODIFYPERM) represent an additive factor $O(S)$ in the overall complexity of the algorithm. Finally, all operations on the set of rules and the superiority relation performed by conditions in lines 5, 12, and 21 require constant time, given the implementation of data structures proposed above. Therefore, we can state that the complexity of the algorithm is $O(S)$. \square

Theorem 31. *Given a Defeasible Theory D with size S , Algorithm 5 COMPUTEDEFEASIBLE terminates and its computational complexity is $O(S)$.*

Proof. When referring to the termination of Algorithm 5 COMPUTEDEFEASIBLE, the most important part we have to analyse is the **repeat** cycle in lines 7–35. Once an instance of the cycle has been performed, we must be in one of the following (mutually exclusive) situations:

1. no modification of the extension has occurred. In this case, line 29 ensures the termination of the algorithm;
2. the theory is modified with respect to a literal in the Modal Herbrand Base HB . Notice that the algorithm takes care of removing the literal from HB once it has performed the suitable operations. As the set is finite, the process described above eventually empties HB , and at the next iteration of the cycle we have no means to modify the extension of the theory. In this case as well, the algorithm ends its execution.

The analysis of the complexity of Algorithm 5 COMPUTEDEFEASIBLE straightly follows from the reasoning proposed to demonstrate the computational complexity of Algorithm 4 CHECKFACTS. Thus, the **repeat** cycle in lines 7–35 is in $O(S)$, while procedure invocations in lines 12, 17, 19, 26 and 28 represent an additive factor as before. Since the

operations in lines 15 and 24, and the checks in lines 14, 16, 18, 23, 25 and 27 also weight a constant time, the computational complexity of Algorithm 5 COMPUTEDEFESIBLE is bound to $O(S)$. \square

Theorem 32. *Algorithm 5 COMPUTEDEFESIBLE is sound and complete.*

Proof. As already argued at the beginning of the section, the aim of Algorithm 5 COMPUTEDEFESIBLE is to compute the defeasible extension of a given theory D through successive transformations on the set of facts and rules, and on the superiority relation. These transformations act in ways which obtain at each step a new simpler theory while retaining the same extension. Again, we remark that the word ‘simpler’ is used to denote a theory with less elements in it. Since we have already proved the termination of the algorithm, it eventually comes to a fix-point theory where no more operations can be made.

In order to demonstrate the soundness of Algorithm 5, we show in the list below that all the operations performed by the algorithm are those described in Lemmas 19–28, where we have already proved the soundness of the operation involved.

- Algorithm 1 DISCARD:
 - Lines 2–4: Lemma 21 and 23.
- Algorithm 2 MODIFYOBL:
 - Lines 2–4, 10–13: Lemma 20;
 - Lines 6–9: Proposition 14 and Lemma 20.
- Algorithm 3 MODIFYPERM:
 - Lines 2–3, 9–11: Lemma 22;
 - Lines 5–8: Proposition 13 and Lemma 22.
- Algorithm 4 CHECKFACTS:
 - Lines 3–4: Lemmas 19–23;
 - Lines 5–8: Lemma 19;
 - Lines 9–11: Algorithm 2;
 - Lines 12–17: Lemma 21;
 - Lines 18–20: Algorithm 3;
 - Lines 21–25: Lemma 23.
- Algorithm 5 COMPUTEDEFESIBLE:
 - Lines 11–13: Lemma 27;
 - Lines 17 and 26: Lemma 28;
 - Lines 18–20: Lemmas 25 and 20;
 - Lines 27–29: Lemma 26 and 22.

These results state that if in the initial theory a literal is either defeasibly proved or not, so it will be in the final theory; thus proving the soundness of the algorithm.

Moreover, since all lemmas show the equivalence of the two theories, and since the equivalence relation is a bijection, this also gives the completeness of Algorithm 5 COMPUTEDEFESIBLE. \square

6. Discussion: The Three Types of Permission

Resuming the discussion of Section 2, we will delve into the technical aspects of the three mentioned concepts of permissions within our framework.

The idea of weak or negative permission is easily represented in the system as follows:

Definition 33 (Weak Permission). *Let D be a Defeasible Theory. A literal l is weakly permitted in D iff $D \vdash -\partial_O \sim l$.*

One remark is in order here: Definition 33 is useful to check whether a literal l is weakly permitted within the theory, but it cannot be directly used to explicitly derive Pl for triggering any rule where this modal literal occurs in the antecedent. In fact, when Pl appears in the antecedent of a rule, then the only way to activate such a rule is to explicitly derive Pl .

However, weak permissions are decisive in the applicability of a rule for conditions (1.2) of Definitions 5 and 6; when $\neg Ol$ occurs in the antecedent of the rule, then the theory must prove $-\partial_O l$. This is equivalent to say that $\sim l$ is weakly permitted in D .

This reading assumes that the distinction between weak and strong permission goes beyond the idea, defended by [2], that there is only one prescriptive type of permission. If the reader finds our proposal limiting, we can trivially revise the rule applicability conditions at point (1.3) (and adjust the algorithms accordingly) by establishing that, when Pl occurs in the antecedent of any rule r , r is applicable if one of the following conditions holds: (i) $+\partial_P l$, or (ii) $-\partial_O \sim l$ (observe that $+\partial_P l$ implies $-\partial_O \sim l$, but not vice versa).

A straightforward result (from Proposition 15 Part 1) regarding weak permissions follows:

Proposition 34. *Let D be any O -consistent Defeasible Theory. For any literal l , if $D \vdash +\partial_O l$, then l is weakly permitted.*

As expected, weak permission enjoys the deontic principle “Ought implies Can”, i.e., the principle that in deontic logic is $Ol \rightarrow Pl$.

We now consider the two ways to obtain strong permissions in Defeasible Logic: using either explicit permissive norms or defeaters.

The first case is naturally captured in the logical framework proposed in Sections 3 and 4. In the simplest case, a literal like Pl is derived in a theory D when there is a successful reasoning chain in which the last rule has the form $a_1, \dots, a_n \Rightarrow_P l$.

More complex cases are due to the fact that l may occur in an \odot -expression. In this case l is obtained as strongly permitted if, for each literal c preceding l in the sequence, one of the following conditions hold:

- if c leads to derive Oc (i.e., c occurs in an \otimes -subsequence of the main sequence where l occurs), then this obligation must be obtained and violated;
- if c leads to derive Pc (i.e., c occurs in an \odot -subsequence of the main sequence where l occurs), then this permission is successfully attacked by an opposite obligation.

The introduction of sequences of permissions and obligations enriches the language in a significant way, since it allows us to express a preference among obligations and permissions when they are logically compatible. In the case of sequences of positive per-

missions, an \odot -sequence states that a permissive exception of an obligation is preferred with respect to another possible exception of the same obligation. However, this extension does not conceptually change the fundamental intuition that is also behind the basic case where permissive norms have the form $a_1, \dots, a_n \Rightarrow_P l$: the antecedent of positive permissive rules with head l provides sufficient defeasible reasons to derive Pl .

The second method considered in Section 2 to capture the notion of strong permissions acting as exceptions to obligations looks at permissions as *undercutters* in argumentation theory (this idea was discussed in [10]): a permissive norm with head l operates in such a way that, if applicable, it is not a sufficient reason for deriving neither l , nor $\sim l$, but it is a sufficient reason for blocking the derivation of $\sim l$ as obligatory. In Defeasible Logic, this idea is naturally implemented by using defeaters. For the sake of simplicity, we have not considered this concept of strong permission in Sections 3 and 4. However, to cover this case it is sufficient to adopt one of the following definitions (compare the definition of $R^P[q, n]$ in Section 3):

Definition 35. *The set $R^P[q, n]$ is $X \cup Y$ where*

- *X is the set of rules where q appears at index n , and the operator \odot precedes q ;*
- *Y is the set of defeaters with head q .⁷*

Definition 36. *The set $R^P[q, n]$ is the set of defeaters with head q .⁸*

Definition 35 adds the defeaters to the set of rules that can be used to derive tagged literals like $+\partial_P l$, obtaining modal literals like Pl . Definition 36 restricts derivations of strong permissions only to reasoning chains where the last rule is a defeater. In both cases, except these new definitions, we do not need to change anything else in the logic (hence, in the proof conditions for $\pm\partial_P$) or the algorithms.

What is the difference between strong permissions obtained via rules for permission and the ones obtained via defeaters? Although rules for P and defeaters are not in general equivalent, as we have informally suggested in Section 2, they behave quite similarly when they are used to derive permissions, as well as to attack obligation rules supporting the opposite conclusion. In other words, defeaters do not clash with any permissive rules. Consequently, if this reading of defeaters is simply embedded within the proof conditions for $\pm\partial_P$ and for $\pm\partial_O$ by adopting either Definition 35, or Definition 36, then rules for P and defeaters play a very similar role in the proof theory. In fact, if we consider condition (2.3.3) in the proof condition for $\pm\partial_O$, then two rules like $r_1 : a_1, \dots, a_n \Rightarrow_P l$ and $r_2 : a_1, \dots, a_n \leadsto l$ both attack any rule s for obligation supporting $\sim l$, and s can counterattack r_1 and r_2 as well.

We remark that the significant difference between the rules for P and defeaters is that defeaters do not allow for having sequences of permissions in their head.

Finally, notice that neither type of strong permission considered enjoys the principle “Ought implies Can”. This result comes directly from Proposition 15 and is based on the idea that the only manner to derive strong permissions is by means of reasoning chains where the last rule occurring in them is either a rule for P or a defeater (i.e., explicit permissive norms).

⁷In this case, n is always 1.

⁸Since n is always 1, n can be omitted and we can simply write $R^P[q]$.

7. Summary and Related Work

In this paper we proposed an extension of Defeasible Logic to represent three concepts of defeasible permission. In particular, we examined different types of explicit permissive norms that work as exceptions to opposite obligations. We also discussed how strong permissions can be represented with or without introducing a new consequence relation for inferring conclusions from explicit permissive norms. Finally, we combined a preference operator applicable to contrary-to-duty obligations with a new one representing ordered sequences of strong permissions which derogate from prohibitions. Special attention was devoted to the computational aspects of the logic.

Although logicians have mostly overlooked the concept of permission over time, the history of deontic logic offers some well-known key ideas to interpret it. Indeed, the original intuition (proposed by [34], among others) that permissions are the modal dual of obligations, though technically simple and attractive, proved to be partial and simplistic (for a discussion, see [35,2,1]). Hence, subsequent contributions have helped to expand the picture in several directions. The distinction between weak (or negative) and strong (or positive) permission [35] plays an important role in this regard. Though, Alchourrón and Bulygin [3,2] argued that there is only one prescriptive sense of permission, while the distinction between weak and strong permission makes sense only at a descriptive level, depending on how any permission is obtained within a system of norms. Legal theorists such as Alf Ross and Norberto Bobbio [28,8] claimed that legal permissions are in fact exceptions to obligations imposing the opposite, even though this did not lead them (Ross, in particular) to clearly link the concept of exception with the one of strong permission. Other theorists even denied the usefulness of seeing strong permissions as exceptions [27,29], since the former ones introduce nothing but strong indifference in normative systems. This thesis was instead rejected by [3].

The purpose of this paper was not to provide a comprehensive logical theory of permission, nor to take an exhaustive critical position in the debate that we have very briskly recalled. Our goal was twofold:

- to capture some aspects of permissions within a broader view of defeasible normative reasoning;
- to study the defeasibility of permissions in a computationally efficient logical framework.

At a more theoretical level, our work shares with [25,9,33] some conceptual assumptions. Slightly rephrasing [33]’s analysis, the following guidelines inspired our treatment of the concept of permission:

1. “no logic of norms without attention to the system of which they form part” [26]: our investigation of the concept of permission looks at how permissive norms and other types of norms interact within systems;
2. the distinction between positive and negative permission is meaningful;
3. one fundamental role of positive permissions is that of stating exceptions to obligations; accordingly, positive permissions are supposed to override, or at least block, some deontic conclusions coming from other norms;
4. the logical space of weak permission is the one left unregulated by mandatory norms.

However, [25,9,33] are all based on a different logical formalism, Input/Output Logic (IOL) [24], thus it is difficult to compare in detail those contributions with the present

one. Normative reasoning is viewed in IOL as a rule-based process of manipulation of inputs (factual premises) into outputs (normative conclusions). The analysis of normative systems consists in representing conditional norms simply as ordered pairs (a, b) where a represents the antecedent of the rule, and b its consequent: “if a then b ” where a has factual content and b normative content, viz. an obligation or a permission. Typically, both a and b are taken to be formulas from propositional logic. Each set of such ordered pairs can be seen as an inferential mechanism which, given an input, determines an output based on those connections. Formally, given a set of positive mandatory norms (obligations) G and a set of permissive norms (positive permissions) P , a closure operation C , and a set of facts A , the output of $G \cup P$ given C and a set of input formulas is $out_C(G \cup P, A) = \{b \mid (a, b) \in C(G \cup P) \text{ and } s \in A\}$. This approach allows for defining different concepts of permission [25,9]⁹:

Negative permission: (a, x) is a negative permission w.r.t. G iff $(a, \neg x) \notin out_C(G)$; if x is not prohibited by the system given a , then x is negatively permitted under those factual conditions a .

Static permission: (a, x) is a static permission w.r.t. (G, P) iff $(a, x) \in out_C(G \cup \{(c, d)\})$ for some $(c, d) \in P$; (a, x) is statically permitted iff it follows from adding a positive permissive norm to G ;

Dynamic permission: (a, x) is a dynamic permission w.r.t. (G, P) iff $(c, \neg d) \in out_C(G \cup \{a, \neg x\})$ for some $(c, d) \in P$; (a, x) is permitted when, given the obligations in G , we cannot prohibit x under the condition a without prohibiting d under condition c which is however explicitly permitted by the system.

Another concept of permission was proposed in [33] to specifically capture the idea of exception¹⁰:

Exemption: (a, x) is an exemption w.r.t. (G, P) iff $(a, \neg x) \in out_C(G) \setminus out_C(G) - (c, \neg d)$ for some $(c, d) \in P$; (a, x) is an exception if the code contains the prohibition of x under condition a which, unless it is removed, it clashes with an explicit permission in P .

Since IOL and Defeasible Logic are different logical systems, which were designed for very different purposes, it is difficult to compare them also in regard to the problem of permission. Despite any possible connection between the two logics, which is still an open research problem (Defeasible Logic in fact characterises a consequence relation falling within cumulative reasoning [7]), an immediate comparison shows significant differences. In particular, formulas in IOL are based on propositional logic while rules in Defeasible Logic are built using atomic literals, modal literals, and their negations. A second difference is that the inference machinery of IOL leads to derive pairs, while the inference output in Defeasible Logic refers to theories and consists of sets of tagged literals.

However, there are some general similarities in both approaches.

- First, Defeasible Logic, like IOL, models explicit and implicit permissions by distinguishing in an analogous manner a consequence relation for obligation and one for permission.
- Second, the definition of negative or weak permission in both formalisms is the same.

⁹[33] offers a different technical treatment, which is however in line with most intuitions discussed in [25,9].

¹⁰[33] proposed two definitions. Here, we report on the simpler one.

- Third, although we have not discussed in our approach the notion of static permission, it seems relatively simple to capture it in Defeasible Logic: indeed, we may derive that some p is permitted by making an essential use in the derivation of a rule for explicit permission. The only feature that makes the difference with respect to Defeasible Logic is that in IOL static permission admits the principle “Ought implies Can”, which does not hold for strong permission in our approach.

Similarly, since both approaches distinguish between permissions rebutting obligations and permissions providing exceptions, we do not see any difficulty in capturing in Defeasible Logic the intuition behind the concept of exemption, even though exceptions are more naturally captured in Defeasible Logic by using the superiority relation between rules. The concept of dynamic permission can be instead expressed in Defeasible Logic, but in a different way, due to the sceptical character of Defeasible Logic: if we add a prohibition for some p , which is incompatible with any rule for explicit permission (or even a defeater), then we cannot derive such a prohibition (unless it is stated to be stronger than any other rule), and so p is dynamically permitted.

A novelty of our paper is the introduction of the new operator \odot to express preferences between explicit permissions. A somehow similar idea has been suggested (though with different purposes) by [9], where a preference relation among pairs (for obligations and permissions) was introduced. Technically, it is not clear if that approach can be re-framed in our setting. In fact, adopting that option in Defeasible Logic would not work, as the superiority relation in Defeasible Logic plays a role in the proof theory only in case of rule conflicts. A clear advantage of the current proposal is that it adopts a rich formal language where

- modal operators can occur in the applicability conditions of rules (the inputs in IOL are always factual);
- we have two ordering types between permissions: the one expressed by \odot and the one induced by the superiority relation which applies to defeaters.

To the best of our knowledge, there is no logical system having linear complexity with analogous normative reasoning capabilities.

Acknowledgements

This work is an extended and revised version of the paper presented at Jurix 2011 [16]. We thanks the anonymous referees for their valuable comments.

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy, the Australian Research Council through the ICT Centre of Excellence program and the Queensland Government.

References

- [1] Carlos E. Alchourrón. Philosophical foundations of deontic logic and the logic of defeasible conditionals. In J. J. Meyer and R. J. Wieringa, editors, *Deontic Logic in Computer Science: Normative System Specification*. Wiley, 1993.
- [2] Carlos E. Alchourrón and E. Bulgin. Permission and permissive norms. In W. Krawietz et al., editor, *Theorie der Normen*. Duncker & Humblot, 1984.
- [3] Carlos E. Alchourrón and Eugenio Bulgin. The expressive conception of norms. In Risto Hilpinen, editor, *New Studies in Deontic Logic*, pages 95–125. D. Reidel Publishing Company, Dordrecht, 1981.

- [4] Robert Alexy. On balancing and subsumption. a structural comparison. *Ratio Juris*, 16(4):433–449, 2003.
- [5] Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. Representation results for defeasible logic. *ACM Trans. Comput. Logic*, 2:255–287, April 2001.
- [6] Grigoris Antoniou, David Billington, Guido Governatori, Michael J. Maher, and Andrew Rock. A family of defeasible reasoning logics and its implementation. In *ECAI 2000*, pages 459–463, 2000.
- [7] David Billington. Defeasible logic is stable. *J. Log. Comput.*, 3(4):379–400, 1993.
- [8] N. Bobbio. *Teoria della norma giuridica*. Giappichelli, 1958.
- [9] G. Boella and L. van der Torre. Permissions and obligations in hierarchical normative systems. In *Proceedings of the 9th international conference on Artificial intelligence and law*, ICAIL ’03, pages 109–118, New York, NY, USA, 2003. ACM.
- [10] G. Boella and L. van der Torre. Permissions and undercutters. In *Proceedings of the IJCAI-03 Workshop on Nonmonotonic Reasoning, Action, and Change (NRAC’03)*, pages 51–57, Acapulco, 2003.
- [11] Guido Boella and Leendert van der Torre. Permission and authorization in normative multiagent systems. In *Proceedings of the 10th international conference on Artificial intelligence and law*, ICAIL ’05, pages 236–237, New York, NY, USA, 2005. ACM.
- [12] M.A. Brown. Conditional obligation and positive permission for agents in time. *Nordic Journal of Philosophical Logic*, 5(2):83–111, 2000.
- [13] E. Bulgin. Permissive norms and normative systems. In A. Martino and F. Socci Natali, editors, *Automated Analysis of Legal Texts*. Publishing Company, 1986.
- [14] G. Governatori and A. Rotolo. Logic of violations: A gentzen system for reasoning with contrary-to-duty obligations. *Australasian Journal of Logic*, 4:193–215, 2006.
- [15] G. Governatori and A. Rotolo. Justice delayed is justice denied: Logics for a temporal account of reparations and legal compliance. In *CLIMA XII, 12th International Workshop on Computational Logic and Multi-Agent Systems*. Springer, 2011.
- [16] Guido Governatori, Francesco Olivieri, Antonino Rotolo, and Simone Scannapieco. Three concepts of defeasible permission. In Katie M. Atkinson, editor, *Legal Knowledge and Information Systems - JURIX 2011: The Twenty-Fourth Annual Conference*, volume 235 of *Frontiers in Artificial Intelligence and Applications*, pages 63–72. IOS Press, 2011.
- [17] Guido Governatori, Vineet Padmanabhan, Antonino Rotolo, and Abdul Sattar. A defeasible logic for modelling policy-based intentions and motivational attitudes. *Logic Journal of the IGPL*, 17(3):227–265, 2009.
- [18] Guido Governatori, Antonino Rotolo, and Erica Calardo. Possible world semantics for defeasible deontic logic. In Thomas Ågotnes, Jan Broersen, and Dag Elgesem, editors, *DEON*, volume 7393 of *Lecture Notes in Computer Science*, pages 46–60. Springer, 2012.
- [19] Guido Governatori, Antonino Rotolo, and Giovanni Sartor. Temporalised normative positions in defeasible logic. In *ICAIL 05*, pages 25–34. ACM Press, 2005.
- [20] Guido Governatori and Shazia Sadiq. The journey to business process compliance. *Handbook of Research on BPM*, pages 426–454, 2008.
- [21] Guido Governatori and Sidney Shek. Rule based business process compliance. In *Proceedings of the RuleML2012@ECAI Challenge*, number 874 in CEUR, page 5, 2012.
- [22] Ho-Pun Lam and Guido Governatori. What are the Necessity Rules in Defeasible Reasoning? In James Delgrande and Wolfgang Faber, editors, *LPNMR-11*, pages 187–192. Springer, 2011.
- [23] Michael J. Maher. Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming*, 1(6):691–711, 2001.
- [24] D. Makinson and L. van der Torre. Input-output logics. *Journal of Philosophical Logic*, 29(4):383–408, 2000.
- [25] D. Makinson and L. van der Torre. Permission from an input/output perspective. *Journal of Philosophical Logic*, 32(4):391–416, 2003.
- [26] David C. Makinson. On a fundamental problem of deontic logic. In Paul McNamara and Henry Prakken, editors, *Norms, Logics and Information Systems. New Studies in Deontic Logic and Computer Science*, pages 29–54. IOS Press, Amsterdam, 1999.
- [27] K. Opalek and J. Wolenski. Normative systems, permission and deontic logic. *Ratio Juris*, 4:334–348, 1991.
- [28] A. Ross. *Directives and norms*. Routledge and Kegan Paul, 1968.
- [29] Lambèr M. M. Royakkers and Frank Dignum. The logic of enactment. In *ICAIL*, 1997.

- [30] G. Sartor. *Legal Reasoning: A Cognitive Approach to the Law*. Springer, 2005.
- [31] Giovanni Sartor. Doing justice to rights and values: teleological reasoning and proportionality. *Artif. Intell. Law*, 18(2):175–215, June 2010.
- [32] A. Stolpe. Relevance, derogation and permission. In *DEON*, pages 98–115. Springer, 2010.
- [33] A. Stolpe. A theory of permission based on the notion of derogation. *J. Applied Logic*, 8(1):97–113, 2010.
- [34] G.H. von Wright. Deontic logic. *Mind*, 60(237):1–15, 1951.
- [35] G.H. von Wright. *Norm and action: A logical inquiry*. Routledge and Kegan Paul, 1963.

A. Appendix

We now prove the properties related to operations that transform a theory D into an equivalent simpler theory D' (we recall that the term “simpler” means a theory with a minor number of symbols in it). The transformations operate on rules either by removing some elements from some rules, or by deleting rules from a theory. Given the functional nature of the operations involved, we will refer to the rules in the target theory with the same names/labels as the rules in the source theory. Thus, given a rule $r \in D$, we will refer to the rule corresponding to it in D' , if it exists, with the same label, namely r .

Lemma 19. *Let $D = (F, R, >)$ be a theory such that $p \in F \cap \text{Lit}$. Let $D' = (F', R', >')$ be the theory obtained from D , where*

$$\begin{aligned}
 F' &= F \setminus \{p\} \\
 R' &= \{r : A(r) \setminus \{p\} \Rightarrow_{\text{O}} C(r) \mid p \in R, A(r) \cap \tilde{p} = \emptyset\} \cup \\
 &\quad \{r : A(r) \setminus \{p\} \Rightarrow_{\text{P}} C(r) \mid r \in R, A(r) \cap \tilde{p} = \emptyset\} \\
 >' &= > \setminus \{(r, s), (s, r) \mid r, s \in R, A(r) \cap \tilde{p} \neq \emptyset\}.
 \end{aligned}$$

Then $D \equiv D'$.

Proof. The proof is by induction on the length of a derivation P .

For the inductive base, we consider all the modal derivations for a generic literal q in the theory.

$P(1) = +\partial_{\text{O}}q$. This is possible in two cases: (1) $\text{O}q \in F$, or (2) $\text{O}\sim q \notin F$, $\neg\text{O}q \notin F$ and $\text{P}\sim q \notin F$, and $\exists r \in R^{\text{O}}[q, i]$ that is applicable for q at i at $P(1)$ and every rule for $\sim q$ is either (a) discarded for $\sim q$ at $P(1)$, or (b) defeated by a stronger rule for q applicable at $P(1)$.

For (1), by construction of D' , $\text{O}q \in F$ iff $\text{O}q \in F'$, thus $+\partial_{\text{O}}q$ is provable both in D and in D' .

For (2), again by construction of D' , the modal literals in the clause do not appear in F iff they do not appear in F' . Furthermore, an obligation rule $r \in R^{\text{O}}[q, i]$ is applicable for q at $P(1)$ iff $i = 1$, $A(r) \subseteq F$, and $\sim p \notin A(r)$ since $p \in F$. Therefore $A(r) \subseteq F$ iff $A(r) \setminus \{p\} \subseteq F'$. This means that if a rule is applicable at $P(1)$ in D then it is applicable at $P(1)$ in D' . In the other direction, suppose that r is applicable in D' , thus in particular $A(r) \subseteq F'$. In both cases where r has p in its antecedent or it is not in D , we obtain $A(r) \subseteq F$, therefore r is applicable in D .

Let us now consider a rule attacking r , namely a rule $s \in R[\sim q, j]$. For such a rule, we have to analyse cases (a) and (b) above.

(a) A rule s is discarded for $\sim q$ at j at $P(1)$ in D iff: (i) $\exists a_i \in A(s) \cap \text{Lit}$ and $a_i \notin F$, or (ii) $\exists c_k \in C(s)$, $k < j$ such that $c_k \in F$, and $s \in R^O[c_k, k]$ by condition (2) of a rule being discarded for $+\partial_O$.

For (i), we are sure that $a_i \neq p$ since $p \in F$ by hypothesis. If $a_i = \sim p$, then s is discarded in D and, by construction, the rule is not in D' . Hence $R'[\sim q] \subseteq R[\sim q]$. If $a_i \neq \sim p$, then by construction $a_i \notin F$ iff $a_i \notin F'$. For (ii), if $c_k = p$, then the rule is discarded in D , the consequent of s is truncated at k in D' , and $\sim q$ does not occur in the consequent of s in D' , i.e., $s \notin R'[\sim q]$. If $c_k \neq p$, then the rule is also discarded in D' since the only difference between F and F' is that p is in F but not in F' . To summarise, if a rule is discarded for $\sim q$ at j at $P(1)$ in D , then the rule is either not in D' , or discarded in D' .

For the other direction, in R' there are no rules containing either p , or $\sim p$. Hence, if we have $a_i \in A(s)$ and $a_i \notin F'$, then $a_i \notin F$. Similarly for c_k , if $c_k \in F'$, then $c_k \in F$. The difference between D and D' is that in R we have rules with $\sim p$ in the antecedent and rules with p preceding q in the consequent, and these rules are not in R' . Since $p \in F$, all rules in R for which there is no corresponding rule in R' are discarded in D .

(b) The superiority relation is modified in a way that we only remove instances where one of the rules is discarded in D . But only rules that are not discarded are active in the clauses of the proof conditions where the superiority relation is involved.

From the discussion above, if a rule is applicable for q at 1 at $P(1)$ in D , then the rule is also applicable in D' . If a rule is discarded for $\sim q$ at index 1 at $P(1)$ in D , then the rule is not in D' , or it is discarded in D' . If a rule s for $\sim q$ is applicable in D , then there is an applicable rule t for q stronger than s . The rules s and t are applicable, so they are in D' and $t >' s$. Thus, $D' \vdash +\partial_O q$.

Similarly to the other direction, if a rule is applicable in D' then it is applicable in D , and if it is discarded in D' then it is discarded in D . The additional rules in D have either $\sim p$ in the antecedent, or p in the consequent, thus these rules are discarded in D , and for them clause (2.3.1) of $+\partial_O$ applies. Therefore if we have a derivation of length one of $+\partial_O q$ in D' , then we have $D \vdash +\partial_O q$.

$P(1) = +\partial_P q$. The proof is essentially identical to the inductive base for $+\partial_O q$, with some slight syntactical modifications dictated by the different proof conditions for $+\partial_P$: (1) $Pq \in F$, or (2) $O\sim q \notin F$ and $\neg Pq \notin F$, and $\exists r \in R^P[q, 1]$ that is applicable for q at $P(1)$ and every obligation rule for $\sim q$ is either (a) discarded for $\sim q$ at $P(1)$, or (b) defeated by a stronger rule for q applicable at $P(1)$.

$P(1) = -\partial_O q$. Clearly clauses (1) and (2.1) of $-\partial_O$ hold for D iff they hold for D' , given that F and F' have the same modal literals. For clause (2.2), consider a rule $r \in R^O[q, 1]$. If r is discarded for D then, as we argued above, it is not in R' , or it is discarded in D' . Also, all rules discarded in D' are discarded in D , and all rules in R for which there is no corresponding rule in R' are discarded in D as well. As regards clauses (2.3.1)–(2.3.3), we point out that condition $t \not>' s$ between two applicable rules t and s is clearly unaffected passing from D to D' , and the other way around.

$P(1) = -\partial_P q$. This case is a mere variant of the previous one for the negative provability of obligations.

For the inductive step, the property of equivalence between D and D' is assumed up to the n -th step of a generic proof for a given literal l .

$P(n+1) = +\partial_O q$. Clauses (1) and (2.1) of $+\partial_O$ follow the same conditions treated in the inductive base for $+\partial_O q$. As regards clause (2.2), if an applicable rule $r \in R^O[q, i]$ for q in D exists, then clauses (1.1)–(1.5) and (2) of Definition 5 are all satisfied. By inductive hypothesis, we conclude that clauses (1.1)–(1.4) are satisfied by r in D' as well, and clause (1.5) is satisfied in D' by the inductive base. For condition (2), the provability of c_k as an obligation in D' is given by inductive hypothesis; furthermore, $c_k \notin F$ or $\sim c_k \in F$ iff $c_k \notin F'$ or $\sim c_k \in F'$ either when $c_k = \sim p$ or $c_k \neq \sim p$ since F and F' coincide in both cases (notice that $c_k \neq p$ by hypothesis).

The direction from rule applicability in D' to rule applicability in D is straightforward. Therefore, a rule $r \in R^O[q, i]$ is applicable for q in D iff it is applicable for q in D' . Conditions (2.3.1)–(2.3.3) are treated like cases (a) and (b) for the corresponding inductive base.

$P(n+1) = +\partial_P q$. Again, the inductive base justifies clauses (1), (2.1), (2.3.1), and (2.3.2) of $+\partial_P q$. Clause (2.2) is satisfied by the same reasoning used in the inductive step of $+\partial_O q$ and by Definition 6, whose additional clause (3) is trivially satisfied by inductive hypothesis.

$P(n+1) = -\partial_O q$. Besides conditions (1), (2.1), (2.3.1), and (2.3.2) – treated as usual – it remains to prove that a rule for q as a permission at $P(n+1)$ is discarded in D iff it is discarded at $P(n+1)$ in D' . To this end, we follow the same analysis carried out in $P(n+1) = +\partial_O q$ for rule applicability, using the inductive base and hypothesis, and Definition 7.

$P(n+1) = -\partial_P q$. This case is a mere variant of the previous one for the negative provability as an obligation. \square

Lemma 20. *Let $D = (F, R, >)$ be a theory such that $F \cap \text{Lit} = \emptyset$ and $D \vdash +\partial_O p$. Let $D' = (F, R', >')$ be the theory obtained from D where*

$$\begin{aligned} R' &= \{r : A(r) \setminus \{Op\} \Rightarrow_O C(r) ! \sim p \ominus \sim p, \\ &\quad A(r) \setminus \{Op\} \Rightarrow_O C(r) \ominus p \mid r \in R, A(r) \cap \widetilde{Op} = \emptyset\} \cup \\ &\quad \{r : A(r) \setminus \{Op\} \Rightarrow_P C(r) \ominus \sim p \mid r \in R, A(r) \cap \widetilde{Op} = \emptyset\} \\ &>' = > \setminus \{(r, s), (s, r) \mid r, s \in R, A(r) \cap \widetilde{Op} \neq \emptyset\}. \end{aligned}$$

Then $D \equiv D'$.

Proof. The proof is by induction on the length of a derivation P .

For the inductive base, we consider all possible derivations of length one for a generic literal q in the theory, where $Op \in F$.

$P(1) = +\partial_O q$. From D to D' , the structure of the proof follows the inductive base for $+\partial_O q$ of Lemma 19, where the cases depending on F are trivially satisfied since $F = F'$, and the other steps are obtained by substituting p with Op and $\sim p$ with \widetilde{Op} .

From D' to D , there must exist an applicable rule r proving $+\partial_O q$ at $P(1)$ in D' . Then $A(r) \subseteq F$. By construction of D' , the antecedent of r in D is either the same, or $A(r) \cup \{Op\}$, while the consequent has either q as the first element, or only p precedes q . Since $Op \in F$ and $p \notin F$, all the combinations of antecedent and consequent denote applicable rules in D .

As already argued, also in this case if a rule s is discarded at $P(1)$ in D , then it is not in D' , or it is discarded in D' . In particular, all rules in R for which there is no corresponding rule in R' have either (i) $\neg Op$, $O\sim p$ or $P\sim p$ in the antecedent, or (ii) $\sim p$ in the consequent. Since $+\partial_O p$ holds, clause (1.2) of Definition 7 and Proposition 15 Parts 1 and 2 make the rules of the form (i) discarded in D . Rules of type (ii) are also discarded in D since $Op \in F$.

$P(n+1) = +\partial_O q$. The proof is essentially identical to situation $P(n+1) = +\partial_O q$ of Lemma 19. Notice that for rule applicability, clauses $l \in F$ (condition (1.5)), and $c_k \notin F$ or $\sim c_k \in F$ (condition (2)) are both true in D and D' , since the set of facts is the same and it does not contain non-modal literals.

$P(1) = +\partial_P q$, $P(n+1) = +\partial_P q$. For the inductive base, notice that by construction of D' , the antecedent of a rule r for permission in D is either the same, or $A(r) \cup \{Op\}$, while the consequent has either q as the first element, or only $\sim p$ precedes q . However, applicability and refutability of this rule follow the analysis carried out for $+\partial_O$. We treat the inductive step as usual, using the inductive hypothesis and the fact that $F = F'$.

The hypothesis $F = F'$ and the structure of D' can be easily used to prove the inductive base and the inductive step for proof tags $-\partial_O$ and $-\partial_P$.

Both the inductive base and the inductive step for $\pm\partial_X q$ with $X = \{O, P\}$ in the case where $Op \notin F$ are straightforward. Indeed, even when $Op \in A(r)$ in D , the hypothesis $+\partial_O p$ allows an applicable rule in D' to be also applicable in D . The same hypothesis allows us to conclude that a discarded rule in D is also discarded in D' , and the other way around. \square

Lemma 21. *Let $D = (F, R, >)$ be a theory such that $F \cap \text{Lit} = \emptyset$ and $D \vdash -\partial_O p$. Let $D' = (F, R', >')$ be theory obtained from D where*

$$\begin{aligned} R' &= \{r : A(r) \setminus \{\neg Op\} \Rightarrow_O C(r)!p \ominus p \mid r \in R, A(r) \cap \{Op\} = \emptyset\} \\ &>' = > \setminus \{(r, s), (s, r) \mid r, s \in R, A(r) \cap \{Op\} \neq \emptyset\}. \end{aligned}$$

Then $D \equiv D'$.

Proof. Since $D \vdash +\partial_O l$ implies $D \vdash -\partial_O \sim l$ by Proposition 15 Part 1, the modifications on R' and $>'$ represent a particular case of Lemma 20, where $l = \sim p$. The only difference is that we eliminate from D rules with Op in the antecedent, and we modify the antecedent of the others eliminating $\neg Op$ (we recall that condition $-\partial_O p$ makes $\neg Op$ defeasibly proved in our framework). In the case that $R^O[p] = \emptyset$, no modifications on the consequent of rules are made since literal p does not appear in any chain by hypothesis. \square

Lemma 22. *Let $D = (F, R, >)$ be a theory such that $F \cap \text{Lit} = \emptyset$ and $D \vdash +\partial_P p$. Let $D' = (F, R', >')$ be theory obtained from D where*

$$\begin{aligned} R' &= \{r : A(r) \setminus \{Pp\} \Rightarrow_O C(r)! \sim p \ominus \sim p \mid r \in R, A(r) \cap \widetilde{Pp} = \emptyset\} \cup \\ &\quad \{r : A(r) \setminus \{Pp\} \Rightarrow_P C(r)!p \mid r \in R, A(r) \cap \widetilde{Pp} = \emptyset\} \\ &>' = > \setminus \{(r, s), (s, r) \mid r, s \in R, A(r) \cap \widetilde{Pp} \neq \emptyset\}. \end{aligned}$$

Then $D \equiv D'$.

Proof. The proof is by induction on the length of a derivation P ; its structure is the same of that for Lemma 20. We have to take into account the different proof conditions for permission, and arrange the proof to analyse the inductive bases and steps of each derivation either when Pp is a fact or not.

For the inductive base, we consider all possible derivations of length one for a generic literal q in the theory, where $Pp \in F$.

$P(1) = +\partial_O q$. From D to D' , the structure of the proof follows the inductive base for $+\partial_O q$ of Lemma 19, where the cases depending on F are trivially satisfied since $F = F'$, and the other steps are obtained by substituting p with Pp and $\sim p$ with \widetilde{Pp} .

From D' to D , there must exist an applicable rule r proving $+\partial_O q$ at $P(1)$ in D' . Then $A(r) \subseteq F$. By construction of D' , the antecedent of r in D is either the same, or $A(r) \cup \{Pp\}$, while the consequent has q as the first element. In this case p cannot precede q since a permission never precedes an obligation in a reparation chain. Since $Pp \in F$, then all rules in D corresponding to applicable rules in D' are themselves applicable in D .

As already argued, also in this case if a rule s is discarded for $\sim q$ at $P(1)$ in D , then either it is not in D' , or it is discarded in D' . In particular, all rules in R for which there is no corresponding rule in R' have either (i) $\neg Pp$ or $O\sim p$ in the antecedent, (ii) $\sim p$ precedes $\sim q$ if s is an obligation rule, (iii) p precedes $\sim q$ if s is a permission rule. Since $+\partial_P p$ holds, clauses (1.1) and (1.4) of Definition 7, and Proposition 15 Part 3 make the rules of the form (i) discarded in D . Moreover, the rules of type (ii) and (iii) are also discarded since $Pp \in F$, and by Definition 7 clauses (2) and (3), respectively.

$P(n+1) = +\partial_O q$. The proof is essentially identical to situation $P(n+1) = +\partial_O q$ of Lemma 19. Notice that for rule applicability, clauses $l \in F$ (condition (1.5)) and $c_k \notin F$ or $\sim c_k \in F$ (condition (2)) are both true in D and D' , since the set of facts is the same and it does not contain non-modal literals.

$P(1) = +\partial_P q$, $P(n+1) = +\partial_P q$. For the inductive base, notice that by construction of D' , the antecedent of a rule r for permission in D is either the same, or $A(r) \cup \{Pp\}$, while the consequent must have q as first element. However, applicability and refutability of this rule follow the analysis carried out for $+\partial_O$. We treat the inductive step as usual, using the inductive hypothesis and the fact that $F = F'$. In this case, we do not consider rules of type (iii), i.e., rules for $P\sim q$, since only rules for obligation can attack rules for permission.

The hypothesis $F = F'$ and the structure of D' can be easily used to prove the inductive base and the inductive step for proof tags $-\partial_O$ and $-\partial_P$.

Both the inductive base and the inductive step for $\pm\partial_X q$ with $X = \{O, P\}$ in the case where $Pp \notin F$ are straightforward. Indeed, even when $Pp \in A(r)$ in D , the hypothesis $+\partial_P p$ allows an applicable rule in D' to be also applicable in D . The same hypothesis allows us to conclude that a discarded rule in D is also discarded in D' , and the other way around. \square

Lemma 23. Let $D = (F, R, >)$ be a theory such that $F \cap \text{Lit} = \emptyset$ and $D \vdash -\partial_P p$. Let $D' = (F, R', >')$ be theory obtained from D where

$$R' = \{r : A(r) \setminus \{\neg Pp\} \Rightarrow_P C(r) \ominus p \mid r \in R, A(r) \cap \{Pp\} = \emptyset\}$$

$$>' = > \setminus \{(r, s), (s, r) \mid r, s \in R, A(r) \cap \{Pp\} \neq \emptyset\}.$$

Then $D \equiv D'$.

Proof. Condition $D \vdash -\partial_p \sim l$ is another consequence of $D \vdash +\partial_O l$, as stated by Proposition 15 Part 2. As such, the proof is derived by Lemma 20, where $l = \sim p$. \square

Lemma 25. *Let $D = (F, R, >)$ be a theory such that $F \cap \text{ModLit} = \emptyset$, $\exists r \in R^O[p, 1]$, $A(r) = \emptyset$, and $R[\sim p] \subseteq R_{\text{infid}}$. Then $D \vdash +\partial_O p$.*

Proof. Given that there are no modal literals in F clause (2.1) of $+\partial_O$ is satisfied. Let r be a rule that meets the conditions of the Lemma. According to Definition 5, rule r is trivially applicable for p in the condition for $+\partial_O$, and thus clause (2.2) is applicable as well. Finally, for clause (2.3) we have that all rules for $\sim p$ are inferiorly defeated by an appropriate rule with empty antecedent for p , but a rule with empty body is applicable. Hence, all clauses for proving $+\partial_O$ are satisfied. Thus, $D \vdash +\partial_O p$. \square

Lemma 26. *Let $D = (F, R, >)$ be a theory such that $F \cap \text{ModLit} = \emptyset$, $\exists r \in R^P[p, 1]$, $A(r) = \emptyset$, and $R^O[\sim p] \subseteq R_{\text{infid}}$. Then $D \vdash +\partial_P p$.*

Proof. The proof is analogous to the previous one. The differences are that we have to use the notion of applicability in Definition 6, and that the rules that are inferiorly defeated are restricted to rules in $R^O[\sim p]$. \square

Lemma 27. *Let $D = (F, R, >)$ be a theory such that $F \cap \text{ModLit} = \emptyset$ and $R^X[p] = \emptyset$, for $X \in \{O, P\}$. Then $D \vdash -\partial_X p$.*

Proof. If there are no modal literals and the set of defeasible (obligation/permission) rules for a literal p is empty, then clause (2.2) of $-\partial_O$ and $-\partial_P$ are vacuously satisfied. \square

Lemma 28. *Let $D = (F, R, >)$ be a theory such that $F \cap \text{ModLit} = \emptyset$, and $\exists r \in R[p, 1]$ such that $A(r) = \emptyset$ and $r_{\text{sup}} = \emptyset$. Then*

1. *if $r \in R^O$, then $D \vdash -\partial_{\square} \sim p$, $\square \in \{O, P\}$;*
2. *if $r \in R^P \cup R_{\text{def}}$, then $D \vdash -\partial_O \sim p$.*

Proof. Let r be a rule in a theory D for which the conditions of the Lemma hold. It is easy to verify that for both cases the rule satisfies clause (2.3) of $-\partial_{\square}$, in particular (2.3.2–3) for $-\partial_O$ and (2.3.2) for $-\partial_P$. \square